Author:
Wallace, Max Ian

Title:
Real-time dynamic substructuring for mechanical and aerospace applications : control techniques and experimental methods

Real-time dynamic substructuring for mechanical and aerospace applications; control techniques and experimental methods.

Max Ian Wallace
Department of Mechanical Engineering

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Doctor of Philosophy in the Faculty of Engineering.

Word Count: 67,500

30^{th} January 2006

Abstract

Real-time dynamic substructuring is an experimental technique for testing the dynamic behaviour of complex structures. It involves creating a hybrid model of the entire structure by combining experimental test piece(s) — the substructure(s) — with a set of numerical model(s) describing the remainder of the system. By employing real-time control techniques to "glue" the numerical and experimental parts together, we create a *virtual* testing environment that if performed correctly will emulate the dynamic behaviour of the *complete* structure exactly.

In this thesis, we focus on the experimental side of substructuring, specifically concentrating on the influence of delays within the substructured system. These are introduced by the inherent dynamics of the actuation device(s) involved — it is impossible for any controlled transfer system(s) (as they are known) to react instantaneously to a state change in demand. We study the stability of the substructured system in direct relation to the magnitude of this delay error and present two methods for identifying the critical limit of stability; firstly, using a delay differential equation approach by approximating the transfer system to a delayed unit response of the numerical model, and secondly, by observing the magnitude of permissible phase margin of the substructured system. We discuss two different formulations of a compensation scheme; one achieving adaptive forward prediction using polynomial extrapolation and the second achieving lag compensation via the inversion of an identified model of the transfer system. We then extend this control strategy to include the concept of robustness which leads us to develop a four stage testing methodology that can be applied to any substructured system to help ensure successful testing.

We build on these fundamental concepts to demonstrate the "proof of concept" of real-time dynamic substructuring for an industrial aerospace application — a helicopter lag damper connected to numerical model of an individual blade excited by flight test data.

ACKNOWLEDGEMENTS

First and foremost, I must thank my parents for their immense love and support and for always having the right words of encouragement when the work was getting tough.

To my supervisor, Dr. David Wagg, who has committed so much time and energy to me and from whom I have learnt so much, I can only begin to show my appreciation. To Dr. Simon Neild, my co-supervisor, you always made me strive to achieve more. And, to all the other esteemed colleges who I had the privilege and the pleasure to work with, I am truly grateful.

To all my boys, thanks for keeping me sane and never letting me die of thirst. And finally, to my beautiful girlfriend, Georgie, for just being her.

AUTHORS DECLARATION

I declare that the work in this dissertation was carried out in accordance with the regulations of the University of Bristol. The work is original except where indicated by special reference in the text and no part of the dissertation has been submitted for any other degree. Any views expressed in the dissertation are those of the author and in no way represent those of the University of Bristol. The dissertation has not been presented to any other University for examination either in the United Kingdom or overseas.

SIGNED:

Max Ian Wallace

30^{th} January 2006

PUBLICATIONS

As a result of the work conducted in this thesis, the following publications have been produced:

Journal Publications:

M.I. Wallace, D.J. Wagg and S.A. Neild. An adaptive polynomial based forward prediction algorithm for multi-actuator real-time dynamic substructuring, *Proc. R Soc. A*, **461**(2064):3807–3826, 2005.

M.I. Wallace, J. Sieber, S.A. Neild, D.J. Wagg and B. Krauskopf. Stability analysis of real-time dynamic substructuring using delay differential equation models, *Earthquake Engng Struct. Dynam.*, **34**(15):1817–1832, 2005.

M.I. Wallace, A. Gonzalez-Buelga, S.A. Neild and D.J. Wagg. Control techniques for real-time dynamic substructuring, *IJAT-aci*, **1**(1):1–3, 2005.

P.J. Gawthrop, M.I. Wallace and D.J. Wagg. Bond-graph based substructuring of dynamical systems, *Earthquake Engng Struct. Dynam.*, **34**(6):687–703, 2005.

P.J. Gawthrop, M.I. Wallace, S.A. Neild and D.J. Wagg. Robust real-time substructuring techniques for under damped systems, *Structural Control and Health Monitoring*, In Press 2006.

Conferences:

M.I. Wallace, D.J. Wagg and S.A. Neild. Use of control techniques for error analysis of real time dynamic substructure testing, *Proc. of the 3rd European Conf. on Structural Control, Vienna, Austria, 12-15 July, 2004*, pp. 59–62.

M.I. Wallace, D.J. Wagg and S.A. Neild. Multi-actuator substructure testing with applications to earthquake engineering: how do we assess accuracy?, *Proc. of the 13th World Conf. Earthquake Engineering, Vancouver, Canada, 1-6 August, 2004*, Paper No. 3241, pp. 1–14.

M.I. Wallace, A. Gonzalez-Buelga, S.A. Neild and D.J. Wagg. Control techniques for real-time dynamic substructuring, *Proc. of IADAT Int. Conf. on Automation, Control and Instrumentation, Bilbao, Spain, 2-4 February, 2005*, pp. 56–60.

M.I. Wallace, S.A. Neild and D.J. Wagg. A stability analysis of substructured systems for real-time dynamic testing, *Proc. of 1st Int. Conf. on Advances in Experimental Structural Engineering, Nagoya, Japan, 19-21 July, 2005*, Paper No. 88, pp. 385–392.

A. Gonzalez-Buelga, D.J. Wagg, M.I. Wallace, S.A. Neild and J.H.G. Macdonald. Testing an autoparametric pendulum using a hybrid numerical experimental method, *Proc. of 1st Int. Conf. on Advances in Experimental Structural Engineering, Nagoya, Japan, 19-21 July, 2005*, Paper No. 62, pp. 417–424.

M.I. Wallace, J. Sieber, S.A. Neild, D.J. Wagg and B. Krauskopf. Delay differential equation models for real-time dynamic substructuring, *Proc. of IDETC/CIE ASME International Design Engineering Technical Conferences, California, USA, 24-28 September, 2005*, Paper No. 85010, pp. 1–8.

Contents

List of Figures

List of Tables

Notation

Chapter 1

Variable	Description
CDM	Central difference method
DDE	Delay differential equation
DOF	Degree-of-Freedom
EFT	Effective Force Testing
EH101	AgustaWestland medium lift helicopter
MCS	Minimal Control Synthesis
MIMO	Multi-Input-Multi-Output
MRAC	Model Reference Adaptive Control
NEES	Network for Earthquake Engineering Simulation
ODE	Ordinary differential equation
OSM	Operator-splitting method
OSM-RTS	Explicit operator-splitting method for real-time testing
PsD	Pseudo-dynamic
RC	Reinforced concrete
SISO	Single-Input-Single-Output
C	Damping vector
F	External force vector
F_e	Elastic restoring force
F_g	Gravity force
F_i	Dynamic inertia force
F_{eff}	Effective force vector
I	Unity matrix
K	Stiffness vector
L	Length
M	Mass vector
N	Number of degrees of freedom
R	Nonlinear restoring force vector
R^E	Explicit nonlinear corrective force vector
R^I	Implicit linear force vector
g	Acceleration due to gravity
x	Relative displacement
$\tilde{x}$	Predicted displacement
x_a	Absolute displacement
x_g	Ground displacement
α	integral shifting parameter

Variable	Description
β	Approximation parameter for the Newmark method
γ	Approximation parameter for the Newmark method
λ	Scaling factor
ρ	Density
v	Velocity
Δt	Sampling time
$\dot{}$	First derivative with respect to time
$\ddot{}$	Second derivative with respect to time

Chapter 2

Variable	Description
2D	Two-dimensional
AC	Alternating current
CPU	Central processing unit
DSP	Digital signal processor
DOF	Degree-of-Freedom
FEA	Finite Element Analysis
LBBP	Linear ball bearings with double lip seals
LVDT	Linear Variable Differential Transformer
MB	Mega byte
DRAM	Dynamic random access memory
ODE	Ordinary differential equation
PID	Proportional + Integral + Derivative control
PsD	Pseudo-dynamic
A	General given matrix
B	General given matrix
C	Stiffness component
D	Damping vector
F	Force fed back from the substructure
$G(S)$	Plant Laplace transfer function
I	Inertial component
K	Stiffness vector
M	Mass vector
S	Excitation vector
R	Damping component
c	Damping constant
e	Global substructuring error
e_1	Numerical model error
e_2	Local control (synchronization) error
$f(.)$	General function
k	Spring constant
k_p	Proportional gain
k_s	Spring constant (substructure)
m	Mass constant
n	Number of states
p	Number of inputs

Variable	Description
r	Excitation state
t	Current time
t_0	Initial time
u	Voltage input to actuator
$u(s)$	Laplace transform of the input from a plant
x	Transfer system state
$y(s)$	Laplace transform of the output from a plant
z	Numerical model state
z'	Delay compensated numerical model state
λ	System eigenvalues
ϕ	State variable
τ	Actual transfer system delay
τ_c	Delayed critical limit
τ_f	Forward critical limit
υ	Velocity
φ	Smallest time constant
ξ	State vector of multi-DOF system
ζ	Damping of closed-loop response
Δt	Sampling time
(t)	With respect to time
$\dot{}$	First derivative with respect to time
$\ddot{}$	Second derivative with respect to time
$(.)^*$	Emulated dynamics
$\rightharpoonup$	Bond symbol
0	Common force junction
1	Common velocity junction

Chapter 3

Variable	Description
2D	Two-dimensional
DDE	Delay differential equation
DOF	Degree-of-Freedom
ODE	Ordinary differential equation
P	Proportional controller
$D(s)$	Transfer function of practical substructured system
$D_0(s)$	$D(s)$ for the case of zero uncertainty
$D_{em}(s)$	$D_0(s)$ for the case of no robustness
F	Force fed back from the substructure
F_e	Equivalent force
F_N	Numerical estimation of feedback force
F_P	Experimentally measured feedback force
$L(s)$	Actual loop gain
$L_0(s)$	Nominal loop gain — the "system"
$N(s)$	Transfer function of *num*
$N_1(s)$	Transfer function representing part of *num*
$N_2(s)$	Transfer function representing part of *num*

Variable	Description
$N_r(s)$	Transfer function between *num* and external forcing
$P(s)$	Transfer function of *phy*
a	Amplitude accuracy of closed-loop response
c	Damping constant
c_{neg}	Negative damping equivalent
d_N	Numerical estimation of numerical model displacement
d_P	Experimentally measured transfer system displacement
e_2	Local control (synchronization) error
j	Complex number
k	Spring constant
k_p	Proportional gain
k_s	Spring constant (substructure)
m	Mass constant
mea	Measurement sensor system
num	Numerical model
p	Ratio of spring stifness
phy	Substructure
r	Excitation state
t	Current time
tra	Transfer system
$\hat{t}$	Non-dimensionalized time
x	Transfer system state
z	Numerical model state
Δ	Robust gain
Δ_d	Robust gain for delay compensation
Δ_l	Robust gain for lag compensation
Λ	Neglected gain — the "controller"
Λ_r	Neglected forward gain
$\Re$	Real part
$\angle$	Phase angle
λ	System eigenvalues
ω	Frequency vector
$\hat{\omega}$	Non-dimensionalized frequency vector
ω_I	Instability frequency substructuring algorithm
ω_n	Natural frequency of the numerical model
ϕ_m	Phase lag
τ	Actual transfer system delay
$\hat{\tau}$	Non-dimensionalized transfer system delay
τ_c	Delayed critical limit
τ_f	Forward critical limit
ζ	System damping
Δt	Sampling time
$\forall$	For all values of
(t)	With respect to time
$\dot{}$	First derivative with respect to time
$\ddot{}$	Second derivative with respect to time
$(.)^*$	Emulated dynamics

Chapter 4

Variable	Description
AFP	Adaptive Forward Prediction
DOF	Degree-of-Freedom
SS	Input/output port of a bond graph
TS	Transfer system
VJ	Virtual Junction
$\hat{F}$	Force synchronization error
F_N	Numerical estimation of feedback force
F_P	Experimentally measured feedback force
F_T	Feedback force estimation for VJ
Mea	Measurement sensor system
N	Order of the polynomial fit
Num	Numerical model
$aNum$	Augmented numerical model
P	Fixed number of time steps to be predicted forward
P_{max}	Maximum number time steps that can be predicted
Phy	Substructure
$aPhy$	Augmented substructure
R	Residual least squares error
T	Transfer function of VJ
Tra	Transfer system
X	Matrix of polynomial x
X_P	Forward prediction vector
a	Variable polynomial coefficients
$\bar{a}$	Fixed polynomial coefficients
c	Damping constant
c_t	Identified equivalent damping
e	Global substructuring error
e_1	Numerical model error
e_2	Local control (synchronization) error
f	Excitation frequency
$f_{predictor}$	Maximum frequency of achievable forward prediction
f_{test}	Excitation frequency of experimental test
k	Spring constant
k_a	Fixed amplitude correction
k_p	Proportional gain
k_t	Identified equivalent stiffness
m	Mass constant
m_t	Identified equivalent mass
n	Number of control points
r	Excitation state
t	Current time
$\hat{v}$	Velocity synchronization error
u	Voltage input to actuator
x	Transfer system state
x_t	Actual displacement for indentification
$\hat{x}_t$	Modelled displacement for indentification

Variable	Description
x_d	Transfer system demand for indentification
y	General polynomial equation
z	Numerical model state
z'	Delay compensated numerical model state
α	Adaptive gain parameter for ρ
β	Adaptive gain parameter for σ
γ	Convergence curve
φ	Trigger state for adaption
ρ	Adaptive number of time steps to be predicted forward
$\bar{\rho}$	Steady state value of ρ
σ	Adaptive amplitude correction
$\bar{\sigma}$	Steady state value of σ
τ	Actual transfer system delay
τ_c	Delayed critical limit
τ_f	Forward critical limit
v_N	Numerical estimation of numerical model velocity
v_P	Experimentally measured transfer system velocity
v_T	Velocity estimation for VJ
Δt	Sampling time
(t)	With respect to time
$\dot{}$	First derivative with respect to time
$\ddot{}$	Second derivative with respect to time
$(.)^*$	Emulated dynamics

Chapter 5

Variable	Description
AFP	Adaptive Forward Prediction
DDE	Delay differential equation
DOF	Degree-of-Freedom
P	Proportional controller
TS	Transfer system
VJ	Virtual Junction
$D(s)$	Transfer function of practical substructured system
$D_0(s)$	$D(s)$ for the case of zero uncertainty
F	Force fed back from the substructure
$L_0(s)$	Nominal loop gain — the "system"
$L_\gamma(s)$	γ compensated loop gain
N	Number of degrees of freedom
$N(s)$	Transfer function of *num*
$N_r(s)$	Transfer function between *num* and external forcing
P	Fixed number of time steps to be predicted forward
$P(s)$	Transfer function of *phy*
$c(s)$	Transfer function for α-compensation
$G_n(s)$	Transfer function of nominal model
$G_a(s)$	Transfer function of the actual system
j	Complex number

Variable	Description
k	Spring constant
k_p	Proportional gain
k_s	Spring constant (substructure)
n	Number of control points
p	Ratio of spring stifness
r	Excitation state
x	Transfer system state
x_m	Nominal model state
z	Numerical model state
Λ	Neglected gain — the "controller"
Λ_r	Neglected forward gain
α	Robustness compensator
γ	Robustness compensator
ζ	Robustness compensator
ζ_r	Magnitude of ζ compensation
ω	Frequency vector
ω_I	Instability frequency substructuring algorithm
ω_n	Natural frequency of the numerical model
ρ	Adaptive number of time steps to be predicted forward
ρ_{max}	Maximum limit of forward prediction
τ	Actual transfer system delay
$\hat{\tau}$	Non-dimensionalized transfer system delay
τ_c	Delayed critical limit
$\hat{\tau_c}$	Non-dimensionalized delayed critical limit
τ_f	Forward critical limit
Δt	Sampling time

Chapter 6

Variable	Description
AC	Alternating current
AFP	Adaptive Forward Prediction
dB	Decibel
DDE	Delay differential equation
DOF	Degree-of-Freedom
DSP	Digital signal processor
EH101	AgustaWestland medium lift helicopter
LVDT	Linear Variable Differential Transformer
NASA	National Aeronautics and Space Administration
PC	Personal computer
PID	Proportional + Integral + Derivative control
PsD	Pseudo-dynamic
R150	Original blade dynamic code form Westland Helicopters Ltd.
R&D	Research and development
RPM	Revolutions per minute
S-G	Savitzky-Golay
A	Cross-sectional area of the piston

Variable	Description
A_1^B	Lateral cyclic control angle constant
B_1^B	Longitudinal cyclic control angle constant
$\underline{F}$	Measured force vector
$\underline{FD}$	Modified forcing of modes in the D-axis
F_{cell}	Load measurement from the load cell
F_{damper}	Actual load on the damper
$G_n(s)$	Transfer function of nominal model
$G'_n(s)$	Reduce accuracy nominal model
I_i^B	Modal inertia
L_D	Absolute distance between the two attachment points of the lag damper
$LDMF_i^{code}$	Constant lag damper modal forcing matrix
$LDMF_i^{exp}$	Measured lag damper modal force
MF_i^{code}	Constant excitation modal forcing matrix
V_d	Damper piston velocity
T_A	Rotation matrix for angle A, where A = β, ζ, θ, γ, δ
$T_i^(j)$	Modal forcing of damper
$\underline{a}$	Vector of global position vector A, where A = A,B,C,D,E
$a_{1,...,6}$	Simplified model coefficients
a_{lc}	Acceleration of the load cell
a_{piston}	Acceleration of the actuator piston
c_1	Idealized damping coefficient for closed blow-off valves
c_2	Idealized damping coefficient for an open blow-off valve
e_2	Local control (synchronization) error
i	Modal number
k_{lc}	Correction factor for added mass
m_{added}	Added mass
n	Number of blades connected to the rotor hub
r	Excitation state
x	Transfer system state
z	Numerical model state
ΔP	Pressure difference between chambers 1 and 2
Ω	Angular velocity
β^B	Angle relative to E
β^B	Flapping angle of the blade
γ	Robustness compensator
γ^B	Angle relative to D
λ_i^B	Modal frequency
ω_D^B	Modal flap deflection
ϕ_i	Modal state
ψ	Azimuth angle
σ	Adaptive amplitude correction
τ	Actual transfer system delay
τ_c	Delayed critical limit
τ_{ca}	Absolute critical limit
θ^B	Angle of blade twist
v_i^B	Structural damping
$\bar{v}_D^B$	Modal lag deflection

Variable	Description
ζ^B	Lagging angle of the blade
0	Centre of hub
(t)	With respect to time
$\dot{}$	First derivative with respect to time
$\ddot{}$	Second derivative with respect to time
Δt	Sampling time

Chapter 7

Variable	Description
CPU	Central processing unit
DDE	Delay differential equation
DOF	Degree-of-Freedom
EH101	AgustaWestland medium lift helicopter
FEA	Finite Element Analysis
ODE	Ordinary differential equation
oe	Output Error
e_1	Numerical model error
e_2	Local control (synchronization) error
n	Number of blades connected to the rotor hub
x	Transfer system state
z	Numerical model state
τ	Actual transfer system delay
Δt	Sampling time

Paginated blank pages are scanned as found in original thesis

No information is missing

Chapter 1

Advanced dynamic testing of structures

SUMMARY: This chapter provides a background to the work presented in this thesis — a comprehensive guide to advanced dynamic structural testing methodologies. We review the main types of laboratory testing of structures under dynamic loads which has led to the development of real-time dynamic substructuring.

1.1 Introduction

The dynamical behaviour of structures has become of great importance to us as a society. We live, work, operate and in the modern world, completely rely on the engineering world around us. We have come to expect many things, including functionality, performance, reliability, aesthetically pleasing designs but most of all, for it to be safe. It is commonly taught to engineering students that "anyone" can make a structure to stand up, but it is the engineer who makes it *just* stand up. In order to do this, we must have a good understanding of all the loading effects, static and dynamic, that the system is likely to face through its lifetime. The term *dynamic* is defined simply as time varying, thus, a dynamic load is any

load for which its magnitude, direction and/or position varies with time. Large structures commonly found in the civil engineering field can experience a whole range of different types of dynamic loading, such as seismic, wind, wave, explosions and impact, human and machine induced vibration, traffic and moving loads [3]. Our understanding of how a structure will respond to such phenomena will effect our ability to produce the most efficient, and therefore "best", engineering design.

There is much literature that has been published on the theoretical analysis of the dynamics of structures, see [4–8]. This thesis is concerned with a very specific type of structural testing, a technique known as *real-time dynamic substructuring*. This field started with the emphasis (clearly) on the simulation of structures under earthquake effects. Predicting the location and intensity of future earthquakes is unfortunately not yet possible. Recent earthquakes such as Kobe (1995) and Umbria (1997) have shown that effective prevention of structural failure should be based mainly on adequate design, construction and maintenance of new civil engineering structures, and retrofitting of existing structures lacking appropriate seismic resistance characteristics. However, the assessment of the seismic vulnerability of structures is a very complex issue due to the non-deterministic characteristics of the seismic motion and the need for an accurate prediction of the structural responses for magnitudes beyond linear behaviour. Satisfactory numerical modeling (typically, the finite element method) depends largely upon the availability of a complete characterization of material properties and appropriate mathematical models able to represent all of the structural components. Chopra [9] provides an insight into dynamical structures with direct application to this field of earthquake engineering. The motivation for the work presented in this thesis is on broadening the application of the real-time dynamic substructuring technique to also being viewed as an advanced form of component testing for mechanical and aerospace systems.

This chapter provides a comparative overview of the current laboratory testing techniques for advanced dynamic structural testing and therefore a background to the work which will be presented in the remainder of the thesis. A recent review of

these technologies is also given by Williams and Blakeborough [10].

1.2 Quasi-static testing

Testing techniques are classified into the *quasi-static* or the *dynamic* category depending on the applied loading rate. For quasi-static testing, a typical test is conducted between tens of minutes and one day. A test specimen is subject to slowly changing pre-prescribed forces or deformations by means of hydraulic actuators. Due to the slow speeds involved, very high load capacity actuators can be utilized such that large structures can be tested. Tests can be conducted either as a monotonic loading test (more commonly known as the *pushover* test) or in a cyclic loading manner. The pushover test is conducted for evaluating maximum strength, maximum displacement, unloading and reloading paths whereas the cyclic loading tests are most commonly used to show the structures's basic hysteretic behaviour [11, 12]. A typical application of quasi-static testing is given by Mehrabi et al. [13] who perform an experimental evaluation of a masonry-infilled reinforced concrete (RC) frame.

Due to the slow loading rate, such that the creep effect and slow bond failure effect become predominant, the internal forces within the structure are not considered in this method. Therefore, the effect of energy dissipation by the dynamic interaction between the structural elements cannot be simulated with sufficient accuracy and thus, the quasi-static experimental techniques cannot capture the dynamic nature of earthquakes. A comparison of the quasi-static and dynamic testing techniques is given by Calvi and Kingsley [14].

1.3 Shaking table testing method

Shaking tables are one of the most important and well established tools available to civil engineers in order to study the effects of seismic motion on structures. They

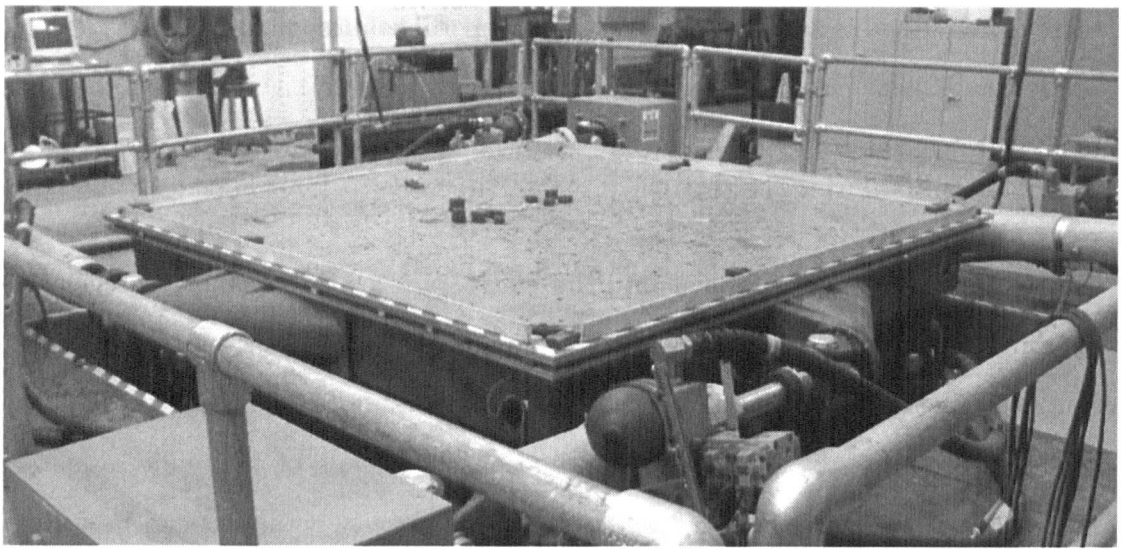


Figure 1.1: BLADE shaking table, EQUALS Lab at the University of Bristol.

were first introduced in the 1940s and became widespread by the 1960s. They consist of a large, stiff platform which is driven by a set of servo-hydraulic actuators (the first tables used profiled cams but this was quickly superseded). The test specimen (also known as the *payload*) is placed on top of the platform and excited according to a predetermined displacement or acceleration control signal. Figure 1.1 shows the new BLADE[1] shaking table situated in the Earthquake Engineering (EQUALS) Laboratory at the University of Bristol. There are four actuators which provide longitudinal and lateral motion (conventionally named, X_1, X_2, Y_1 and Y_2) and four which provide vertical motion (conventionally named, Z_1, Z_2, Z_3 and Z_4). This results in a six Degree-of-Freedom (DOF) system (three-axis translation, roll, pitch and yaw). The actuators outnumber the number of degrees of freedom so that the system is over-constrained, thus only certain configurations of the actuators are possible which are prescribed using a kinematic model.

Shaking tables work by reproducing the inertial forces experienced during an earthquake on the structure in a *real-time* dynamic test (a description of real-time and what this entails is given in § 2.4.4). Essentially, this means that if the control of

[1]Bristol Laboratory for Advanced Dynamics Engineering.

the table can be done accurately all the damping and inertial characteristics of the payload are retained. The control reference to the table (typically an acceleration) is determined by the specific earthquake dynamics and also by the scale and characteristics of the payload. It may be necessary (in fact, likely) to scale down a structure to fit within the table's operational capacity, this introduces *scaling effects* which are discussed in § 1.3.1.

There are many existing shaking table facilities around the world; these can be classified by their absolute size and load capacity and their dynamic rating. The BLADE shaking table for example (Figure 1.1) is relatively small in size, 3m × 3m platform (cast aluminium inverted pyramid weighing 6.5 t), with eight 70kN capacity actuators. However, the actuators have an extremely high dynamic rating with a top speed of 3m/s and thus can reproduce the most demanding of earthquake signals. Shaking tables are not only constrained by capacity but also by cost; construction, operation and maintenance. Some extremely large and expensive facilities have been constructed recently in Japan in an attempt to avoid the problems associated with scaling of the nonlinear dynamic responses. The largest of which (and currently the largest in the world) is in "Miki City" situated in Hyogo Prefecture to the north of the city of Kobe and has a capacity of 1,200 t (equivalent to a four-story reinforced concrete building) and is known as *E-Defense* for short. The extremely high capacity is achieved at the expense of reducing the number of controlled degrees of freedom to only a three-dimensional test, see Ogawa et al. [15]. However, given its size of 15m × 20m, experimental structures can be tested at full scale. The shaking table has five longitudinal actuators, five lateral, and fourteen vertically. It uses servo-hydraulic valves which are able to provide oil flow rates of up to 15,000l/min. The first experiment was carried out in January 2005 and reproduced the earthquake that was recorded at the Kobe Marine Observatory at the time of the Great Hanshin-Awaji Earthquake on a full-scale two-story wooden house.

In control terms, the shaking table can be seen as the "plant" — a physical system which produces an output to a given input. Under closed-loop control, the specific

control algorithm attempts to supply the right input to the table in order to achieve a desired output. The difference between the reference and the output is known as the *synchronisation* error. The main goal of the controller is to minimize the magnitude of this error in order to obtain a faithful representation of the original earthquake.

Two main control approaches can be implemented: the plant can be considered as a Multi-Input-Multi-Output (MIMO) system or as a set of Single-Input-Single-Output (SISO) systems. The first is quite complex, since it considers the table as a whole and has to take the interactions between actuators into account. The second is more simple, as it controls each actuator in isolation. If each actuator follows its own demand exactly, the table will achieve the target reference. However, the influence of noise, computational and transducer errors as well as the dynamic interactions between the table and the payload prevent the plant from reaching the target accurately. This is due to the fact that in shaking table control the dynamics of the system are partly determined by the characteristics and regime of the payload, which is usually of a larger mass than the table itself. The payloads regime often enters into the plastic region and its characteristics are subjected to sudden changes, especially if a collapse occurs during the experiment. During the early 1990s a Pan-European network of shaking table facilities, the ECOEST consortium, was created ([16]). A programme of tests were carried out to compare the performance of the shaking tables; a strong interaction between the payload and the shaking table was discovered and is summarized in Crewe [17]. This interaction can be reduced after a process of fine tuning the control parameters of the table and/or iteratively correcting the input signal (matching), but this is specific to each shaking table and to each specimen and thus is limited.

Many of early shaking tables were fitted only with linear controllers, usually implemented in analog form. A linear controller (also known as a *fixed-gain* controller) is easy to implement, it is stable as long as certain conditions are met and the technology is well known. Linear controllers can also be tuned with a high degree of

precision when the plant has well known parameters and behaves linearly. However, these type of controllers are not well-suited for nonlinear regimes and cannot respond to unexpected plant dynamics or if the plant has been badly estimated. One possible way of overcoming this limitation is to use an adaptive controller; one which the gains are updated at every sampling interval to account for dynamic changes in the system being controlled. Recent attempts have been based around using *model reference* type controllers — of the form taken by the Model Reference Adaptive Control (MRAC) algorithm of Landau [18]. In this scheme, a linear controller is tuned according to the parameters of the plant. The parameters are estimated on-line, comparing the output of the plant to the output of a linear reference model. Both the plant and the linear reference model use the same reference signal. Specifically this has been done at the University of Bristol using the Minimal Control Synthesis (MCS) algorithm [19, 20]. It is well-suited to this application since it requires no prior identification of the dynamics of the system being controlled and can be retrofitted around existing controllers to achieve MIMO control, as described in Stoten and Gómez [21] and Gómez [22].

1.3.1 Scale model testing

The shaking table test has the advantage of retaining the true dynamic characteristics of the earthquake, however it suffers from the fact that it can (usually) only be performed on reduced-scale models of the original structure [23]. Making a reduced scale model which has the exact dynamic structural behaviour representative of the real-scale structure is unfortunately not possible. Dynamic similitude is a technique used to express the similarity of forces between the structure and its corresponding model under excitation, and can be conveniently expressed using two parameters. The Cauchy number is the ratio between the dynamic inertia forces F_i and the elastic restoring forces F_e,

$$\frac{F_i}{F_e} = \frac{\rho v^2}{E}, \tag{1.1}$$

where ρ is density, v is velocity and E is Youngs modulus. The Froude number is the ratio between the inertia and gravity forces,

$$\frac{F_i}{F_g} = \frac{v^2}{Lg}, \tag{1.2}$$

where L is length and g is the acceleration due to gravity. In most cases it is desirable that both the Cauchy and Froude numbers of the model match the values for the structure [16]. The most important consequences of this matching are that the mass-scale factor should be the inverse of the length-scale factor and that the time-scale factor should be the square root of the length-scale factor. Thus, if a structure tested is a scaled model, with its size scaled down to λ^{-1} ($\lambda > 1.0$), a ballast must be added to the model, such that its specific mass is increased by λ, and the time-scale must be reduced by $\lambda^{\frac{1}{2}}$ to reproduce the strain-rate that would be induced in the original structure. However, by adding such a ballast, the axial stress imposed to the model becomes λ times the axial stress for the original structure (axial force is one of the primary factors that lead the structure to failure) and by shortening the time-scale, the resultant increase in the frequency content of the demand signal increases the demands on the actuators. This demonstrates the difficultly in maintaining the structural properties in a reduced-scale model perfectly. The lack of confidence provided the motivation for the development of the large Japanese shaking tables described earlier.

However, the problem becomes more complex when we consider the fabrication of the model itself. In steel structures for example, the thermal energy inputed by welding is much larger per unit volume in a reduced-scale model than it would be in the original structure, which tends to significantly alter the stress distribution as well as material properties of the reduced-scale model. Additionally, the size of the weld itself is much larger increasing the stiffness of the model. As reported in Nakashima [24], when using reduced-scale models for scale-sensitive problems, such as connections in steel structures or bond and anchorage in reinforced concrete, at most the global behaviour of the structure (such as the force-deflection relationship before serious deterioration in resistance) can be simulated, the local behaviour

(such as local buckling, crack propagation and bond deterioration) is beyond the scope. There is no question about the importance of local behaviour in assessing the seismic performance of structures, in many cases structural damage is triggered by local defects such as connection failure or bond deterioration. This leads to the conclusion that if information on local behaviour of the structure is required, the test specimen must be fabricated at full-scale. If this is beyond the capacity of the shaking table another testing technique is required, such as the effective force testing method (§ 1.5), the pseudo-dynamic testing method (§ 1.6) or the hybrid testing method (§ 1.7).

1.4 Dynamics of a structure subjected to an earthquake

We use here the simple example of a planar structure, as can be seen in Figure 1.2, subjected to an earthquake where all the degrees of freedom N are in the same

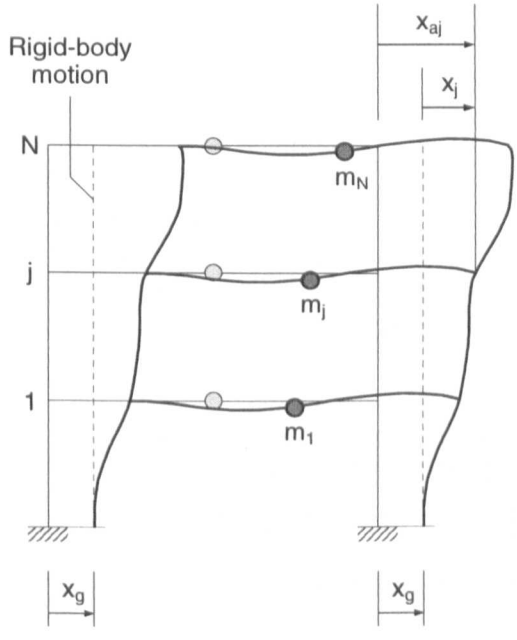


Figure 1.2: Planar structure subject to ground excitation x_g; Relative motion of each mass m_j is given by x_j and the absolute motion by x_{aj}, where $j = 1, \ldots, N$.

direction as the ground motion. From Chopra [9], the equations of motion for such a structure subjected to a ground displacement of x_g can be written as

$$\boldsymbol{M}\ddot{\boldsymbol{x}}_{\boldsymbol{a}} + \boldsymbol{C}\dot{\boldsymbol{x}} + \boldsymbol{K}\boldsymbol{x} = \mathbf{0}, \tag{1.3}$$

where, $\mathbf{M}, \mathbf{C}, \mathbf{K}$ are the matrices of masses, damping and stiffness respectively (and may have a diagonal structure), $\mathbf{x}$ is the vector of the relative displacements and $\boldsymbol{x}_{\boldsymbol{a}}$ is the vector of the absolute displacements given by

$$\boldsymbol{x}_{\boldsymbol{a}} = \boldsymbol{x} + \boldsymbol{I}x_g, \tag{1.4}$$

where $\boldsymbol{I}$ is a unity vector of order N. Substituting Eq. (1.4) into Eq. (1.3) gives

$$\boldsymbol{M}\ddot{\boldsymbol{x}} + \boldsymbol{C}\dot{\boldsymbol{x}} + \boldsymbol{K}\boldsymbol{x} = -\boldsymbol{M}\boldsymbol{I}\ddot{x}_g. \tag{1.5}$$

Eq. (1.5) implies that the structure can be analysed as a structure that is supported on a fixed foundation and subjected to an external force vector, such that $\boldsymbol{F} = -\boldsymbol{M}\mathbf{I}\ddot{x}_g$. Additionally, if the restoring forces, represented by the term $\boldsymbol{K}\boldsymbol{x}$ are replaced by a more general restoring force vector $\boldsymbol{R}$ (which can include nonlinearities), the equations of motion become

$$\boldsymbol{M}\ddot{\boldsymbol{x}} + \boldsymbol{C}\dot{\boldsymbol{x}} + \boldsymbol{R} = \boldsymbol{F}. \tag{1.6}$$

Eq. (1.6) can now be used as the basic equation of motion for the different testing methods described in the remainder of this chapter.

1.5 Effective force testing method

The concept of the Effective Force Test (EFT) method is that the response of a system to a given ground motion may be replicated by applying the external force vector $\boldsymbol{F}$ (known in this case as the *effective* force vector $\boldsymbol{F}_{\boldsymbol{eff}}$) of Eq. (1.6) to the test structure. This is achieved by operating actuators under force control influencing the structure in real-time. Although this is not a new concept [25], the implementation of EFT is still relatively new [26, 27]. As noted in the development of Eq. (1.6),

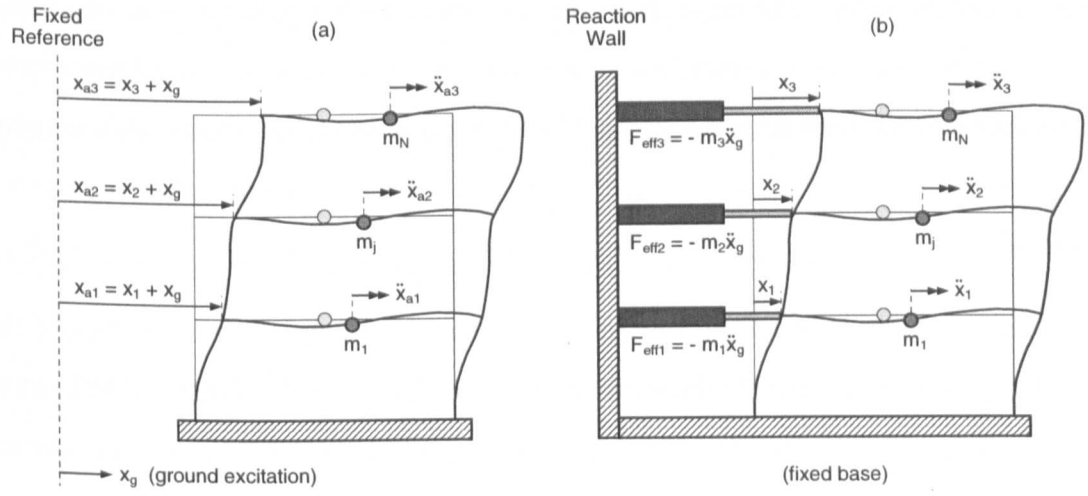


Figure 1.3: Effective force testing: (a) structure and (b) laboratory test set-up.

the effective force at each degree of freedom is equal to the product of the mass and ground acceleration in the direction of the degree of freedom. The concept of the EFT method is illustrated in Figure 1.3 for a multi DOF test structure. Actuators reacting off of a *reaction wall* are utilized to apply the effective load to the test structure.

The key advantage of the EFT method is that the effective forces depend on only the ground acceleration record and the structural mass, and are independent of any nonlinear behaviour of the structure such as stiffness and damping. They can therefore be calculated in advance of the test, and the need for online computations during testing is minimal.

The challenge of using the EFT method is to achieve accurate force representation on the test structure. To simulate the real-time effects of an earthquake on a structure, dynamic actuators and a high quality servo-hydraulic control system are needed to accurately apply the effective forces. Dyke et al. [28] found that there is an intrinsic property of hydraulic actuators, called natural velocity feedback, which restricts the ability of the actuators to apply an accurate force when the test structure is vibrating near one of its natural frequencies. Dimig et al. [29] developed a method

called natural velocity feedback negation to correct for the phenomenon associated with natural velocity feedback. This method is based on classical control theory and was successfully demonstrated for single DOF systems. Zhao et al. [30] extended this technique to encompass nonlinear compensation for nonlinear real-time dynamic testing.

The main disadvantage of the EFT method is that the complete seismic mass of the prototype structure must be included in the test structure. This may be difficult to achieve in all but the largest laboratories. Additionally, only a lumped mass system can be tested.

1.6 Pseudo-dynamic testing method

The pseudo-dynamic (PsD) test method (also known as online testing) is an experimental technique for simulating the response of structures and structural components to earthquake excitation. Unlike conventional direct integration algorithms, such as quasi-static testing § 1.2, in PsD testing the restoring forces are not modelled but measured directly from the test specimen.

According to Nakashima [24], the original concept was proposed by Hakuno et al. [31] in the late 1960s and was not established in its current form, combined with quasi-static loading test, until Takanashi et al. [32] in 1975. The technique continued to be developed under the US-Japan Cooperative Earthquake Program in the 1980s [33–36]. Summaries of these efforts are provided in Takanashi and Nakashima [37] (1987) (for the Japanese activities), Mahin et al. [25] (1989) (for US activities) and Shing et al. [38] (1996) (for combined Japanese and US efforts). A number of specific challenges arose through these developments including problems of experimental errors leading to growth of erroneous responses, problems of scaling, problems with rate-of-loading effects and problems associated with integration algorithms which did not ensure unconditional stability. Many studies were carried out to overcome these problems which broadened its development and application into Europe and

Asia [39–45]. Although it became the typical situation in PsD to control to a displacement, research has been carried out in mixed-state control to reduce the error propagation at low amplitudes by Pan et al. [46] who used a mixed displacement-force algorithm.

One of the critical prerequisites for conducting the PsD test is the effect of the loading rate on the restoring force. Depending on the rate the test structure is being loaded, PsD testing can be divided into two categories, either *conventional* or *real-time*. Conventional PsD tests are performed with expanded time scales [25], originally approximately one hundred times the actual time scale, which allowed the test structure to be inspected between time intervals in order to ensure the integrity of the test specimen. However, *fast* PsD testing is now more common (between 10 to 100 % of real-time). Structures with load-rate sensitive components are not likely able to have their response to seismic loading accurately captured by the conventional PsD test method as the inertia components of the dynamic characteristic are lost, and therefore should be tested using the real-time PsD test method, § 1.6.1. The higher the rate of testing, the higher the requirements on the equipment (the required oil flow rates, the dynamic rating of actuators and the structural integrity of the test rig to both load and vibration). A potential source of error for the original implementation of PsD testing was the incremental stepwise nature of the test — the current displacement is applied over a short *ramp* period (which is less than the sampling time Δt) and the structure is then held stationary for a *wait* period while measurements are taken until the end of the time step. However, if the structure is yielding, then significant force reductions may occur during this time. To overcome this problem, the wait period must be reduced to zero, resulting in a *continuous* PsD test which has subsequently become widely used [47]. A common method of achieving this is by measuring the experimental forces at small intervals during the ramp period and then using a combined extrapolation-interpolation procedure to realize a smooth transition; details of this are given in Nakashima and Masaoka [48].

In the PsD method of testing, the equations of motion for the structure (i.e., Eq. (1.6)) are solved using either an explicit or implicit direct step-by-step integration method to obtain the response of the structure [49–57]. The mass matrix $\boldsymbol{M}$, viscous damping matrix $\boldsymbol{C}$, and the excitation history $\boldsymbol{F}$ are numerically specified. The step-by-step numerical integration is performed in conjunction with the measured restoring forces $\boldsymbol{R}$ from the test structure and are used as part of the input for the next calculation step. One major difficulty in PsD testing is that results are very sensitive to experimental errors. The error sources of a PsD test are primarily executing errors introduced during the loading process and numerical errors introduced during the direct integration process [34, 35]. In order to avoid complications that may result from iterative procedures during PsD testing, the dynamic integration algorithms employed are normally explicitly represented (for example, the explicit Newmark method). The basic equations of the Newmark method [58] are generally formulated as

$$\boldsymbol{M}\ddot{\boldsymbol{x}}_{\boldsymbol{k+1}} + \boldsymbol{C}\dot{\boldsymbol{x}}_{\boldsymbol{k+1}} + \boldsymbol{R}_{\boldsymbol{k+1}} = \boldsymbol{F}_{\boldsymbol{k+1}}, \tag{1.7}$$

$$\boldsymbol{x}_{\boldsymbol{k+1}} = \boldsymbol{x}_{\boldsymbol{k}} + \Delta t\dot{\boldsymbol{x}}_{\boldsymbol{k+1}} + \Delta t^2[(\frac{1}{2} - \beta)\ddot{\boldsymbol{x}}_{\boldsymbol{k}} + \beta\ddot{\boldsymbol{x}}_{\boldsymbol{k+1}}], \tag{1.8}$$

$$\dot{\boldsymbol{x}}_{\boldsymbol{k+1}} = \dot{\boldsymbol{x}}_{\boldsymbol{k}} + \Delta t[(1 - \gamma)\ddot{\boldsymbol{x}}_{\boldsymbol{k}} + \gamma\ddot{\boldsymbol{x}}_{\boldsymbol{k+1}}], \tag{1.9}$$

where Δt is the integration time interval, the subscript k indicates values at time equal to $k\Delta t$ and β and γ are parameters characterising the approximation strategy. This numerical scheme can be directly implemented for pseudodynamic testing by setting parameter β to be zero. Eq. (1.8) is then simplified as

$$\boldsymbol{x}_{\boldsymbol{k+1}} = \boldsymbol{x}_{\boldsymbol{k}} + \Delta t\dot{\boldsymbol{x}}_{\boldsymbol{k}} + \frac{\Delta t^2}{2}\ddot{\boldsymbol{x}}_{\boldsymbol{k}}, \tag{1.10}$$

in an explicit manner. With $\boldsymbol{x}_{\boldsymbol{k+1}}$ calculated, the structure is then repositioned accordingly.

However, the explicit-type algorithms do not preserve the desired characteristics of stability and accuracy. To overcome this difficulty, Thewalt and Mahin [49] presented an unconditionally stable implicit integration scheme via the introduction of

an analog electrical device without iterative procedures for displacement correction. Bursi and Shing [55] (1996) presented an evaluation of some of the most commonly used implicit time-stepping algorithms for pseudodynamic tests.

Additionally, the Newmark algorithm is dissipative when $\gamma > \frac{1}{2}$ such that the effect of the control errors can be reduced. Therefore numerical damping can be achieved at higher modes, where these errors are most problematic, while minimizing the damping effects on the lower modes. Hilber et al. [59] introduced a modification to the conventional Newmark scheme to achieves this, which was generalised by Bonelli and Bursi [56] (2004). From Eq. (1.7), the α-shifted equation can be written as

$$\boldsymbol{M}\ddot{\boldsymbol{x}}_{\boldsymbol{k+1}}+(1+\alpha)\boldsymbol{C}\dot{\boldsymbol{x}}_{\boldsymbol{k+1}}-\alpha\boldsymbol{C}\dot{\boldsymbol{x}}_{\boldsymbol{k}}+(1+\alpha)\boldsymbol{R}_{\boldsymbol{k+1}}-\alpha\boldsymbol{R}_{\boldsymbol{k}}=(1+\alpha)\boldsymbol{F}_{\boldsymbol{k+1}}-\alpha\boldsymbol{F}_{\boldsymbol{k}}, \quad (1.11)$$

This is then solved using Eq. (1.8) and Eq. (1.9) for $-\frac{1}{3} \leq \alpha \leq 0$ with the parameters β and γ given by

$$\begin{aligned} \beta &= \tfrac{1}{4}(1-\alpha^2), \\ \gamma &= \tfrac{1}{2}(1-2\alpha). \end{aligned} \quad (1.12)$$

When $\alpha = 0$ this reduces to the well-known explicit Newmark scheme. As α becomes increasingly negative, the level of numerical damping increases. For all values of α in the stated range, the scheme is implicit. Explicit schemes have the advantage that the required displacement increment can be computed directly from the results of the previous time-step. Implicit methods, however, require knowledge of the acceleration at the end of the current time-step, which can only be achieved by some form of iterative procedure. This is undesirable in structural testing because of the risk of overshooting, which may have a significant effect on the response of the structure.

Nakashima et al. [52] presented an operator-splitting (OS) method by dividing the displacement term into the implicit and explicit parts and is derived under the framework of the aforementioned Newmark method. This has been extended by numerous researchers, such as Combescure and Pegon [57]. The restoring force $\boldsymbol{R}$ in the equation of motion Eq. (1.6) is divided into implicit linear force $\boldsymbol{R^I} = \boldsymbol{K^I x}$

and explicit nonlinear corrective force $\boldsymbol{R^E(\tilde{x})}$, such that

$$\boldsymbol{M\ddot{x}_{k+1} + C\dot{x}_{k+1} + K^I_{k+1} + R^E_{k+1} = F_{k+1}}, \tag{1.13}$$

where $\boldsymbol{K^I}$ is the initial stiffness of the structure, $\boldsymbol{R^E_{k+1} = R_{k+1} - K^I\tilde{x}_{k+1}}$ with $\boldsymbol{\tilde{x}_{k+1}}$ being the predicted displacement of the $(k+1)^{th}$ time instant defined as

$$\boldsymbol{\tilde{x}_{k+1} = x_k + \Delta t\dot{x}_k + (\frac{1}{2} - \alpha)\Delta t^2\ddot{x}_{k+1}}. \tag{1.14}$$

which is the explicit part of Eq. (1.7). During each step of the testing, the structure is positioned according to $\boldsymbol{\tilde{x}_{k+1}}$ estimated by Eq. (1.14) and the restoring force $\boldsymbol{R_{k+1}}$ is measured. The velocity and acceleration responses of the structure are in turn estimated from Eq. (1.8), Eq. (1.9) and Eq. (1.13) and the structure's displacement at time instant $k+1$ is then modified as

$$\boldsymbol{x_{k+1} = \tilde{x}_{k+1} + \alpha\Delta t^2\ddot{x}_{k+1}}. \tag{1.15}$$

If an appropriate parameter is chosen the OS method it is unconditionally stable, as verified by Nakashima et al. [52]. This method is particularly advantageous in testing structures with inelastic behaviour. However, the state-space procedure (SSP) [54] (based on the interpolation of the discrete excitation signals for piecewise convolution integral) is more reliable than the Newmark method in terms of both numerical accuracy and stability. In an attempt to enhance the PsD test, an SSP-based integration algorithm (referred to as the OS-SSP method) is presented via an integration of Nakashima et al. [52] operator-splitting concept with the state-space procedure by Wang et al. [51]. Although the original state-space procedure is unconditionally stable, its derivative (i.e. OS-SSP) does not preserve the desired characteristics of numerical stability, requiring further investigation. However, much of the current research into PsD testing is focusing on achieving real-time control as discussed in the next section.

1.6.1 Real-time pseudo-dynamic testing method

A variety of new types of structural components and devices (such as dampers, rubber bearings and frictional elements) have started to be introduced into structures, particularly in connection with vibration control. Many of these are highly velocity dependent, possessing hysteretic behaviour in their vibrational characteristics and therefore, the conventional PsD test is not suitable. In *real-time* PsD each cycle of the control loop must be completed within the sample time step size, so that the loading and structural response occurs at the same rate in the test as it would in a real dynamic structure. Thus, in a real-time test the force(s) fed back will also include the damping and inertia components (and therefore do not need to be included in the numerical computation Eq. (1.6)) unlike conventional PsD testing where only the static restoring force is fed back. Testing in real-time requires rapid computation and communication, as time-scales are typically in the order of a few milliseconds, but also imposes a number of stringent constraints on the way in which this is achieved (this is discussed in § 2.4.4).

The first real-time PsD test system was developed by Nakashima et al. [60] and applied to a single DOF system loaded by a single actuator. The basic procedures remained the same as those developed for conventional PsD testing, however, a dynamic actuator is now required to replace the quasi-static actuator in order to load the structure with the desired accelerations. The problem with real-time testing is there is only a finite response time of any controlled actuator. Thus, an inevitable delay is introduced between the command signal being sent to an actuator and it moving to the desired position. The force fed back from the experiment is therefore incorrect, since it is measured before the actuator has reached its target position. Nakashima and Masaoka [48] and Horiuchi et al. [61] were the first to observe the need for including a delay compensation scheme in the real-time PsD control algorithm. This was continued by Horiuchi and Konno [62] and has consequently become of fundamental importance to the real-time hybrid testing technique

described in § 1.7 where these compensation techniques continue to be developed. Although stability of an explicit numerical integration scheme can be assured for real-time testing, due to the small sample times involved, the implementation of mixed implicit-explicit algorithms and predictor-corrector numerical schemes have been studied in terms of improving accuracy. Zhang et al. [50] proposed a state-space formulation which is advantageous in real-time PsD testing since most structural control problems are formulated in state space. The development and limitations of real-time PsD testing is discussed in Nakashima [24].

1.7 Hybrid testing method

The hybrid testing method (most commonly known as *substructuring*) is a combined experimental-numerical testing technique. The technique involves creating a hybrid model of the whole system by combining an experimental test piece, known as the *substructure* (in some literature, the *physical substructure*), with one or more numerical models (in some literature, the *numerical substructure*) describing the remainder of the system. The purpose of this is that when it is impractical or impossible to test the dynamic behaviour of an entire system at full scale (and scale or numerical modelling on its own is unreliable, see § 1.3.1) this technique allows the division of the system into these constituent parts. Typically, all the elements (or gross part of the structure) which can be well characterized or evaluated using linear equations are modelled numerically and all the elements which are highly nonlinear and have complex dynamic behaviour are built full size and constrained in a test rig as they would be in the real system. In essence, this creates a "virtual" testing environment for the experimental substructure(s). The principal benefit is that if the testing procedures are carried out correctly then the *virtual* system will mimic the dynamic characteristics of the *complete* system. Therefore, the dynamic properties of the substructure(s) under excitation can then be viewed in terms of its effect on the entire system, rather than the characteristics of their own

dynamic properties in isolation. This technique can therefore provide engineers with a better understanding of the dynamic behaviour of large or complex structures in a laboratory at full scale.

Substructuring has been applied in both PsD and real-time applications, although to date, the PsD side has become more established. This is because the substructuring technique was originally intended to be applied to seismic testing of large structures, directly following on from the conventional PsD tests, described in § 1.6, as the structures had become too large to test practically in a laboratory as discussed in Dermitzakis and Mahin [63]. The first reported case of substructuring was by Nakashima et al. [60] using a viscous damper for the experimental substructure combined with a linear single DOF model of a multi-storey building as the remainder of the structure. This has been followed up in subsequent years by using increasingly complex numerical models and multi-substructure experiments, such as Pinto et al. [64] who performed PsD substructuring tests on a large-scale model of an existing six-pier bridge at the ELSA laboratory, [44, 65]. The two physical pier models were constructed and tested in the laboratory, while the deck, the abutments and the remaining four piers were numerically modeled on-line. However, implementing the substructuring process in real-time, such that the damping and inertial components of the substructure dynamics are retained, introduces its own challenges [1, 66–68], which are similar to those faced in real-time PsD testing (§ 1.6.1). The real-time dynamic substructuring technique is introduced detail in Chapter 2 and discussed throughout the remainder of this thesis, but an overview is given here for completeness.

In real-time substructure testing, constraints are placed on both the numerical and experimental side of the technique. The numerical model computation is explicitly restricted to a single sample time period and the actuation devices used must have a high enough dynamic rating to achieve the desired accelerations. As this currently cannot always be achieved for large structures, real-time substructure testing has been more readily applied to smaller scale component testing, for example Horiuchi

et al. [61], which has led to a broadening of the technique's application away from seismic testing and more towards the field of mechanical engineering as shown in Chapter 6 of this thesis. As only part of the structure is experimentally tested, the nonlinear dynamic behaviour of critical elements or components can be viewed at full scale. Single actuator substructuring has been developed beyond the "proof of concept" stage where experiments on simple substructures have been carried out [62, 66, 69]. Multi-actuator substructuring presents a significant engineering challenge in terms of implementation and transfer system cross coupling [1, 48, 68, 70, 71]. As for real-time PsD testing (§ 1.6.1) an inherent delay is introduced into the substructured system through the control of the actuators. This has far reaching consequences in terms of stability of the substructured system and the control algorithms that must be applied to achieve this.

A number of techniques have been proposed to assess the stability of a substructured system. Horiuchi et al. [61] used an energy analysis of periodic orbits to equate the time delay to a form of negative damping with instability occurring at the point of sign change for the damping of the overall system. Lim et al. [72] examined how the poles of a substructured system, and therefore its stability, are effected by changing the magnitudes of a fixed gain version of an adaptive controller in continuous time. In a similar vein, Darby et al. [73] studied the position of the discrete closed-loop poles of a multi DOF system under the influence of two variable delays. Stability is determined by whether or not the poles lay within the unit circle. Wallace et al. [69] (Chapter 3, § 3.2) employed the method of modelling the substructured system with *delay differential equations* (DDEs), which are derived from the ODE model of the system by explicitly including the delay(s) due to the actuator(s). Gawthrop et al. [74] (Chapter 3, § 3.3) applied tools from control theory to study how the phase margin of the modelled substructured system can provide a measure of how near to instability the ideal system is in terms of how much phase lag is permissible.

In the field of real-time substructuring, the first time delay compensators were obtained by assuming that the dynamics of the transfer system may be approximated

to a pure delay. For example, Horiuchi et al. [61] and Blakeborough et al. [1] proposed outer-loop forward prediction methods which use polynomial extrapolation to predict forward the numerical model displacement by a fixed number of time-steps. Darby et al. [73] relaxed the assumption of a pure delay by developing a forward prediction method that varied the amount of delay compensation, based on the error between the actuator displacement and the desired numerical model displacement. This method was extended by Wallace et al. [75] (Chapter 4, § 4.3.6) who developed an adaptive forward prediction algorithm that used variable polynomial coefficients such that non-integer multiples of the previous time step could be predicted and also incorporates an amplitude correction algorithm. The use of a Smith predictor has also been proposed as a suitable delay compensator Sivaselvan et al. [68]. Lag compensation via an experimental transfer function estimation of the combined inner-loop controller and actuator dynamics has been proposed by Gawthrop et al. [76] (Chapter 4, § 4.4.1) and Sivaselvan et al. [68]. The proposed outer-loop controllers compensate for unwanted dynamics by applying the inverse of the transfer function estimation. Model reference adaptive control has also been suggested as an outer-loop strategy by Wagg and Stoten [67], Neild et al. [77] and Lim et al. [78] which demonstrated how lag compensation can be achieved via this approach.

Gawthrop et al. [74] (Chapter 5, § 5.2) proposed a four stage methodology to achieving a robust substructuring algorithm. The adaptive nature of the outer-loop controllers proposed in [71, 73, 75] allow for the compensation of the induced error despite uncertainty in the dynamics of the actuators. Although they incorporate some level of robustness due to this adaptation, they do not explicitly include a robustness compensator proposed as a separate constituent part. Within the framework of [74] it is possible to combine a number of different types of controller and utilise them as and when they are required.

Traditionally, as the real-time technique typically has a very small sampling time, standard explicit schemes have been used for the integration of the numerical model

as numerical stability can be assured. However, as for the PsD testing technique, the use of different integration algorithms have recently started to be investigated in the context of real-time substructuring. Wu et al. [79] investigated the required modification of the standard central difference method (CDM) due to the fact that in order to obtain the correct velocity dependent restoring force of the substructure, the target velocity is required to be calculated as well as the target displacement. Wu et al. [80] has extended this approach to incorporate the operator-splitting method (OSM) which provides explicit and unconditionally stable solutions for PsD testing technique. However, the OSM only provides an explicit target displacement but not an explicit target velocity, so that it is essentially an implicit method for real-time substructure testing when the velocity-dependent restoring force is considered. Wu et al. [80] proposed a target velocity formulation based on the forward difference of the predicted displacements in order to achieve an explicit OSM algorithm (OSM-RST). The stability and accuracy of the resulting algorithm was investigated using single and multi DOF systems.

1.7.1 Distributed hybrid pseudo-dynamic test method

The George E. Brown Jr. Network for Earthquake Engineering Simulation (NEES) is a current pioneering endeavour in earthquake engineering which started in the 1990s. Funded by the US National Science Foundation, NEES is a research collaboration that connects, through a high performance network, many types of earthquake engineering testing sites located around the world. The NEES testing sites include shaking tables, centrifuges, tsunami/wave tanks, large-scale laboratory experimentation systems, and field experimentation and monitoring installations. All these distributed testing facilities are integrated into a research collaboration by system integration, the *NEESgrid*, which offers a data repository, enables tele-observation and tele-operation, and has capabilities for distributed computer simulation.

Essentially, the concept of the distributed hybrid network brings together the theory

of substructuring (§ 1.7) with the large scale testing of PsD (§ 1.6) and is discussed in detail by Mosqueda [81]. Rather than model any part of the system numerically, the whole structure is divided up into physical substructures and then geographically distributed across the NEES network, taking advantage of the specific facilities in each location. A PsD test can then be performed by passing the relevant information over the internet between each location. Typically, a state flow type controller is utilised such that while there are inevitable delays in passing the information between sites the substructuring test can still be performed continuously, only held in a wait cycle if there is a serious problem to prevent catastrophic damage of the specimens.

1.8 Objectives and scope of current work

The work presented in this thesis is concerned with the development of real-time substructure testing. Specifically, we focus only on the experimental side of the substructuring technique. Although, the accuracy of the numerical modelling techniques (including the stability of the integration strategy used [79, 80]) is an essential part of achieving a substructuring algorithm that can faithfully reproduce the dynamics of the original structure, it is possible, in part, to decouple the two constituent components of this hybrid technique. The real-time architecture that will be used throughout this work, presented in § 2.4.4, allows the use of well known fixed step size explicit algorithms, such as the 4th order Runge Kutta, for which stability can be assured due to the small sample times which will be employed. Additionally, the numerical modelling techniques, described in § 2.4, are only used as tools to facilitate this work. The study of real-time numerical modelling techniques (such as ultra fast finite element analysis) and the stability of specific mixed operator integration algorithms is therefore beyond the scope of this work but a potential direction of future work, § 7.2. The remainder of the thesis is organised into the following structure:

Chapter 2 introduces the concept of creating a substructured system in full. The small scale case study substructuring examples which will be used throughout the remainder of the work are introduced here. Using these conceptually simple systems we are able to introduce the fundamental problems associated with the occurrence of delay in a substructuring algorithm. Additionally, we discuss important concepts from synchronization theory which are essential techniques for understanding the experimental results.

In Chapter 3 we study the stability of the substructured system in direct relation to the magnitude of this delay error present. We present two methods for identifying the critical limit of stability, the maximum value the delay error can reach before the onset of exponentially growing oscillations (the definition of an unstable system). We compute explicit calculations for these stability limits before demonstrating a numerical approach which has the advantage of being suitable for complex and nonlinear systems with more than one delay. It is important to note that this form of instability is a different to the numerical instability encountered from an integration algorithm.

Chapter 4 introduces and discusses the theory of delay compensation in terms of a real-time substructuring system. This is a different principle to the classical control of a delayed system as it is inevitably linked to the stability of the overall algorithm through the magnitude of the delay which remains. We discuss two different formulations of a compensation scheme and show how an adaptive control algorithm can be used to overcome changing plant conditions and any uncertainty in the modelling process for the dynamics of the actuators. Additionally we describe a measure of accuracy for the substructuring algorithm such that the entire system does not have to be modelled.

Chapter 5 extends the control strategy of Chapter 4 to introduce the concept of robustness. Robustness is an essential consideration in the formulation of a successful testing strategy as it reduces the uncertainty of the actuator response

and increases the available margin to the critical limit of stability. However, this is always achieved in compromise of the dynamical accuracy of the numerical model. This work leads us to develop a four stage testing methodology that can be applied to any substructured system to help ensure successful testing.

We build on all the fundamental concepts introduced and developed up to this point in order to study a real industrial problem of substructuring in Chapter 6. We study a helicopter lag damper connected to numerical model of an individual blade excited by flight test data. The damper itself is from the EH101 military utility medium lift helicopter (manufactured by AgustaWestland Ltd.) and is highly complex due to its nonlinear piecewise smooth hysteretic characteristics. We demonstrate the "proof of concept" of real-time substructuring for such a complicated substructured system and show the broadening of the real-time technique from seismic testing of large civil engineering structures to smaller scale component testing of mechanical systems.

Finally, in Chapter 7 we summarize the work presented in this thesis and discuss potential future directions for both the current projects discussed in the thesis and for the field of real-time substructuring itself.

Chapter 2

Real-time dynamic substructuring

SUMMARY: This chapter gives a complete introduction to the real-time dynamic substructuring technique. We discuss the component parts of substructuring and how they can be combined together with a comprehensive control strategy in order to achieve successful experimental implementation. The small scale bench-top case studies, that will be used throughout this thesis, are introduced and described in full in this chapter.

2.1 Introduction

In this thesis we consider the hybrid experimental-numerical testing technique known as *real-time dynamic substructuring*. The technique involves creating a hybrid model of the whole system — the *emulated system* — by combining an experimental test piece — the *substructure* — with one or more numerical models describing the remainder of the system. This creates a "virtual" testing environment for the substructure that, if done correctly, will mimic the dynamic characteristics of the complete system. Therefore, the dynamic properties of the substructure under excitation can then be viewed in terms of its effect on the entire system, rather than the characteristics of its own dynamic properties in isolation. This

technique can therefore provide engineers with a better understanding of large or complex structures when it is impractical to build (or test) the complete system in a laboratory.

The substructuring technique was originally intended to be applied to situations where (accurate) numerical models of experimental parts are unreliable, as discussed in § 1.7. As only part of the entire structure is tested experimentally, it allows engineers to view the behaviour of critical elements under dynamic loading at full scale, thus negating the need to modify to the structure to account for scaling effects as described in § 1.3.1. So far the technique has been developed successfully using expanded time scales, known as pseudodynamic (PsD) testing, see § 1.6 [33, 44, 60, 64]. As PsD testing is carried out quasi-statically, any time-dependent behaviour of the test specimen is lost. However, this type of testing is usually applied to rate-independent structures and as a result has achieved a large amount of success in the field of earthquake engineering because the strain-rate sensitivity of the materials can often be neglected.

Implementing the substructuring process in real-time means that the damping and inertial components of the substructure dynamics are retained [1, 66–68]. To achieve this, constraints are placed on both the numerical and experimental side of the technique. The numerical model computation is explicitly restricted to a single sample time period and the actuation devices used must have a high enough dynamic rating to achieve the desired accelerations. As this currently cannot always be achieved for large structures, real-time testing has more readily been applied to smaller scale component testing, for example Horiuchi et al. [61], which has led to a broadening of the technique's application more towards the field of mechanical engineering as shown in Chapter 6.

To carry out a substructuring test the component of interest is isolated and fixed into an experimental test system. To link the substructure to the numerical models, a set of *transfer systems* (which act on the substructure) are controlled to follow

the appropriate output from each respective numerical model, which is typically a displacement. At the same time the forces between the transfer systems and the substructure are fed back into the numerical models to give a form of bi-directional coupling. Transfer systems are typically single actuators (electric or hydraulic) including their proprietary ("built in") controller, but can also be in the form of a more complex test facility such as a shaking table. The proprietary control forms the *inner-loop* control of a substructuring algorithm and is typically a linear PID controller (software or hardware). Its primary purpose is to achieve a suitable "linear" response from the transfer system such that the effects of uncertainty and nonlinearity are reduced to an acceptable level. Single actuator substructuring has been developed beyond the "proof of concept" stage where experiments on simple substructures have been carried out [62, 66, 69]. Multi-actuator substructuring presents a significant engineering challenge in terms of implementation and transfer system cross coupling [1, 48, 68, 70, 71].

In a substructuring algorithm, the feedback forces (from the substructure) are treated as an external influence (or forcing) on the numerically modelled part of the system, which can then be described by a set of ordinary differential equations (ODEs). This is advantageous as it is simple and fast to integrate an ODE numerically using an implicit or an explicit algorithm, such that real-time control can be achieved as the entire process must take place within the hard real-time constraints. Additionally, if the feedback forces are treated as autonomous entities the dynamics of the numerical models may be decoupled when the substructured system has more than one transfer system, see § 2.3.2. This allows more flexibility in terms of deciding upon which type of control strategy should be employed.

A main focus of the work presented in this thesis is on the fundamental governing principles behind the experimental side of substructuring. The aim of the single and multi degree-of-freedom (DOF) case studies presented in § 2.3.1 and § 2.3.2 is to develop an understanding of the effect of *delay errors* that are always present in a substructured system. Delays arise naturally as it is not possible for any

(controlled) transfer system to react instantaneously to a change of state as prescribed by its numerical model. In fact, there are a number of different delays which combine together to give the overall delay of the transfer system, τ, including data acquisition, computation, digital signal processing and the actuator delay itself. In some situations the transfer system delay may be so small as to be negligible, but the typical situation in substructuring is that this delay is large enough to have a significant influence on the overall dynamics of the substructured system. The loss of stability as a function of increasing delay is typically observed in substructured systems by the onset of oscillations with positive exponential growth. The stability criterion, the maximum (or critical) transfer system delay, τ_c, for a substructured system is discussed in detail in Chapter 3.

It is therefore an essential condition that the *actual* transfer system delay, τ, is less than the *critical* delay, τ_c, in order to ensure that the substructured algorithm is stable. In fact, there are a number of differing states the substructuring algorithm can reside in as can be seen from Table 2.1. Note that in Table 2.1 there are two critical delays; τ_c is defined as the *delayed* critical limit and τ_f is defined as the *forward* critical limit. In order to achieve any of these states a delay compensation scheme must be utilized in the substructuring algorithm. Typically, this has been achieved by including additional control algorithms to act as an *outer-loop* controller around the existing proprietary control. A number of substructuring control strategies have

State	Condition
$\tau > \tau_c$	Unstable algorithm.
$0 < \tau < \tau_c$	Stable: Numerical model(s) has a bounded error.
$\tau = 0$	Stable: Numerical model(s) replicates emulated system exactly.
$\tau_f < \tau < 0$	Stable: Numerical model(s) has a bounded error.
$\tau < \tau_f$	Unstable algorithm.

Table 2.1: Substructuring algorithm states (assuming zero transfer system amplitude error); a *delay* gives positive $+\tau$, a *lead* gives negative $-\tau$.

been reported [1, 61, 67, 68, 71, 73, 75, 76, 78, 82]. These strategies explicitly eliminate the delay introduced by the transfer system in the force fed back from the substructure in various different ways. The outer-loop controllers suggested to date broadly fall into two categories, those that provide (time) delay (e^{-sT}) compensation and those that provide (frequency dependent) lag (e.g. $\frac{1}{1+sT}$) compensation. To date, much of the work has aimed to completely negate the effect of the transfer system dynamics such that $\tau = 0$ and the substructured system "exactly" replicates the dynamics of the emulated system; the delay compensation schemes used in this thesis are presented in Chapter 4. In fact, the *forward* critical limit can also be a useful tool in achieving a more robust substructuring algorithm and is discussed in detail in § 5.5 and in Wallace et al. [75]. It should be noted that the delayed critical limit, τ_c, is solely a function of the system dynamics where as the forward critical limit, τ_f, is defined either by the system dynamics or (more likely) by the characteristics of the delay compensation scheme employed.

Once a stable substructuring algorithm has been achieved for an experimental test, a key measure is that of accuracy [70, 81]. It is the typical situation in substructuring that the dynamics of the substructure are not known in full, in fact, that is the primary reason for performing the substructure test. If this is the case, how do we know if our substructuring test is giving us an accurate portrayal of the components dynamics in situ? In § 2.6 [75] we discuss a substructuring test's accuracy in terms of the component parts of a substructuring algorithm listed below:

***Local* (control) error:** This is the synchronisation error between the *actual* state (typically a displacement) of the transfer system compared to the *desired* state of the numerical model.

***Numerical model* error:** This is the error between *actual* state of the numerical model compared to the *ideal* state of the of the emulated system.

***Global* (substructuring) error:** The overall error between the *actual* state of the transfer system and the *ideal* state of the emulated system. This can also be

described by the *actual* state of the substructure compared to the *ideal* state of the of the emulated system.

The *local* error and the *numerical model* error are coupled in the substructuring algorithm and result in the overall *global* error. The nature of this coupling is system dependent and is unknown unless the emulated system dynamics can be explicitly calculated. As this is not the norm for substructuring, the only physical measure of accuracy available is the *local* error — the synchronisation error of the transfer system. In § 4.3.7 we discuss a measure of the accuracy for a substructuring test without having to simulate the emulated system by inferring information from the actuator performance capacity envelope.

2.2 A generalised substructuring algorithm

To carry out a substructuring test, the numerical model and the *experimental* substructure are run in parallel and interact in real-time to emulate the dynamic behaviour of the complete structure. This interaction is achieved through the exchange of information at the interface of the numerical model and the substructure. A generalised representation of a substructuring algorithm is shown in Figure 2.1. Firstly, the displacements (or higher state derivative) at the interface are calculated using the numerical model(s) and imposed via the actuation device(s) (transfer systems) on the physical substructure. Secondly, the force(s) due to imposing these displacements on the substructure are measured and fed back to the numerical model(s) where they are included at the interface for the next time step.

We can formalize this conceptual substructuring algorithm by converting it into a control block diagram showing the *inner-loop* (proprietary) control — to reduce the uncertainty of the transfer system dynamics to an acceptable level — and the *outer-loop* (delay compensation) control — to achieve a stable substructuring algorithm — as shown in Figure 2.2. The detail of the variables shown Figure 2.2 are as follows; state variable for the ground excitation: r, state variable for the Numerical

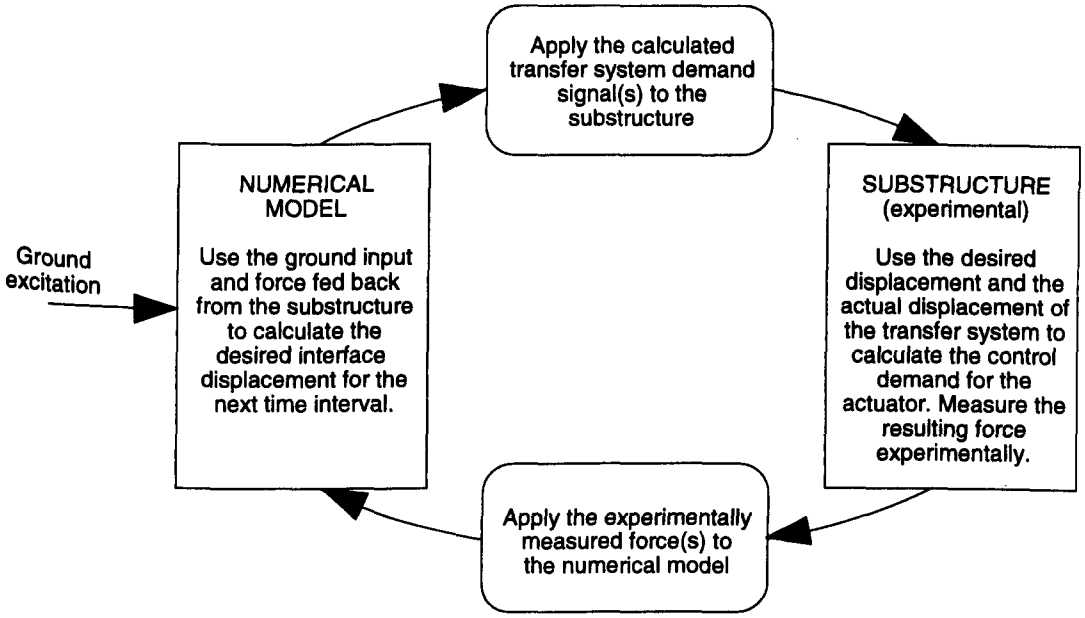


Figure 2.1: A generalised representation of a real-time substructuring algorithm (Adaptation of Blakeborough et al. [1]). The entire cycle must be performed and completed within one sample period Δt.

Model: z, state variable for the delay compensated Numerical Model demand: z', state variable for the Transfer System: x, synchronisation error (local error): e, control input to actuator (voltage): u, resulting force from imposed state change on substructure: F.

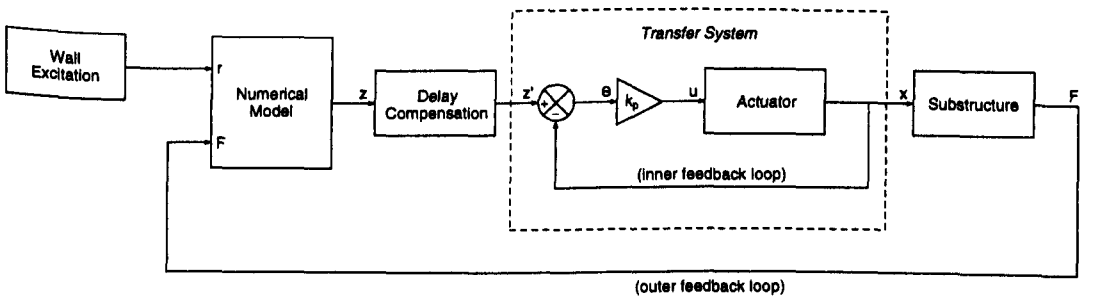


Figure 2.2: A generalised real-time substructuring block diagram (inner-loop control depicted is a linear proportional controller).

It should be noted that from now on the following terminology will be used: z for a *numerical estimation* of a state and x for an *experimentally measured* state. Due to the real-time nature of this type of substructuring the sample time, Δt, is small (typically, no more than a few milliseconds). This allows explicit integration schemes to be used as stability of the numerical schemes can be assured. This is not the case in PsD testing as the time scales are expanded and is one of the fundamental differences in implementing the two techniques. In § 2.4 we discuss the use of integration schemes in more detail.

2.3 Small-scale experimental substructuring case studies

The basis of this thesis is on the fundamental principles behind the experimental side of substructuring. In order to understand these we need to utilise simple examples such that we have full knowledge of the emulated system. In this way it will be possible to test out the ideas presented in this thesis on stability, delay compensation, robustness and accuracy by explicitly measuring the accuracy of the substructuring tests against the ideal dynamics of the emulated system. These ideas can then be developed into an overall strategy, which is presented in § 5.2, for achieving successful large-scale industrial substructuring as described in Chapter 6. The following sections describe both a single and multiple degree-of-freedom (DOF) substructuring case study which will be used throughout the thesis.

2.3.1 Single DOF substructuring example

We consider the example of a single mass-spring oscillator system with one excitation input, shown in Figure 2.3, for the emulated system. This well known linear system will allow us to demonstrate the fundamental problems associated with the occurrence of delay in a substructuring algorithm in its most simple form as there is no inertia in the substructure. The general equation of motion for the system can

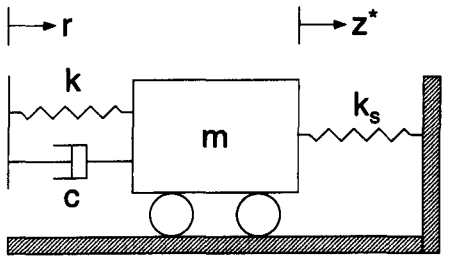


Figure 2.3: Schematic representation of the single mass-spring oscillator

be written as

$$m\ddot{z}^* + c\dot{z}^* + kz^* + k_s z^* = c\dot{r} + kr, \tag{2.1}$$

where, m and c are the mass and damping scalars respectively, k and k_s are the stiffness scalars of the numerical model and the substructure respectively, and r is the support excitation. The state of the system is represented by z^*, where $(.)^*$ is used to indicate that these dynamics are based on the "complete" dynamics of the emulated system and not those of the numerical model of the substructuring algorithm. In this sense, Eq. (2.1) is a good test case; note that for more complex or nonlinear systems it is generally not possible to calculate the emulated dynamics in this way. We can now use Eq. (2.1) to assess the performance of the substructuring algorithm and highlight the effect of the delays in the system.

In order to create a substructured model of the system shown in Figure 2.3, the spring k_s is isolated and taken to be the substructure. The remainder of the structure, the excitation wall and the mass-spring-damper unit, is modelled numerically. This decoupling results in the substructured system shown schematically in Figure 2.4. The dynamics of the numerical model are governed by

$$m\ddot{z} + c(\dot{z} - \dot{r}) + k(z - r) = F, \tag{2.2}$$

where the feedback force F is the substructure response of

$$F = -k_s x, \tag{2.3}$$

(see Figure 2.4). As previously stated, the transfer system has its own dynamics and thus cannot react instantaneously to the change of state of the numerical model.

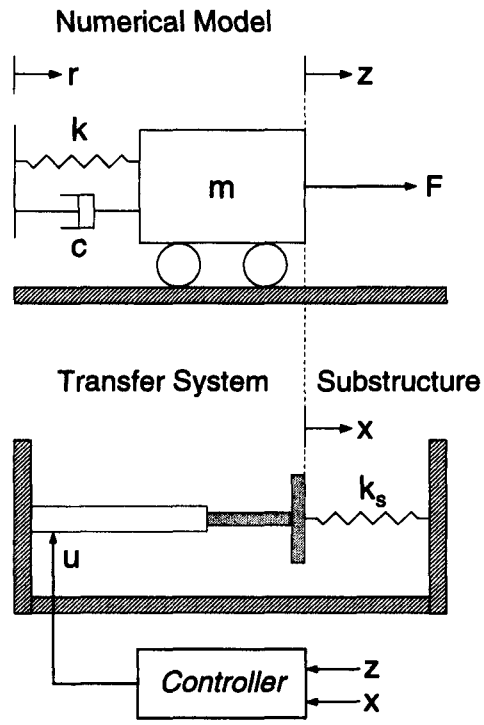


Figure 2.4: Schematic representation of a *substructured* system with one transfer system. State variable for the Numerical Model: $z(t)$, state variable for the Transfer System: $x(t)$, control input to actuator (voltage): $u(t)$.

Accordingly, the inevitable time delay τ is introduced into the transfer system response x and thus, a corresponding error in the feedback force F, as $x \neq z$; the exact nature of this is discussed in Chapter 3. This shows how the feedback force is directly influenced by the transfer system delay and how it introduces a systematic error into the substructuring algorithm. Unless any filtering is performed on the measured force signal (analogue or digital) the delay magnitudes of the transfer system x and the feedback force F will be identical.

2.3.2 Multi DOF substructuring example

The general principle of substructuring remains the same regardless of the number of transfer systems. However, the problem from a control point of view becomes

more complicated for more than a single transfer system due to the introduction of cross-coupling between the control signals. For this reason, we consider the example of a three mass oscillator system with two diametrically opposing excitation walls as shown in Figure 2.5. This will allow us to demonstrate the problems of achieving accurate control for multi-actuator substructuring using a conceptually simple example. The masses are coupled by four linear springs, k_i, and damped by coupled viscous dampers, c_i, where $i = 1, 2, 31, 32$ (constants 31 and 32 indicate firstly they act on the substructure, m_3, and secondly which transfer system they connect to). The system is excited via two moving supports, r_j, where $j = 1, 2$. The general equation of motion for such a system can be written as,

$$\mathbf{M}\ddot{\xi} + \mathbf{D}\dot{\xi} + \mathbf{K}\xi = \mathbf{S}r(t), \tag{2.4}$$

where, $\mathbf{M}$, $\mathbf{D}$ and $\mathbf{K}$ are the mass, damping and stiffness matrices respectively and $\mathbf{S}r(t)$ is the support excitation. $\boldsymbol{\xi}$ is a vector which represents the states of the system, such that

$$\boldsymbol{\xi} = [z_1^*, z_2^*, z_3^*]^T, \tag{2.5}$$

where, $(.)^*$ is again used to indicate that these dynamics are based on the "complete" dynamics of the emulated system. In order to create a substructured model of the system shown in Figure 2.5, the middle mass, m_3, and accompanying springs and dampers (k_{31}, k_{32} and c_{31}, c_{32}) are taken to be the substructure. This leaves both excitation walls and adjacent masses to be used to create two independent numerical models whose influence is imposed on the substructure by two separate

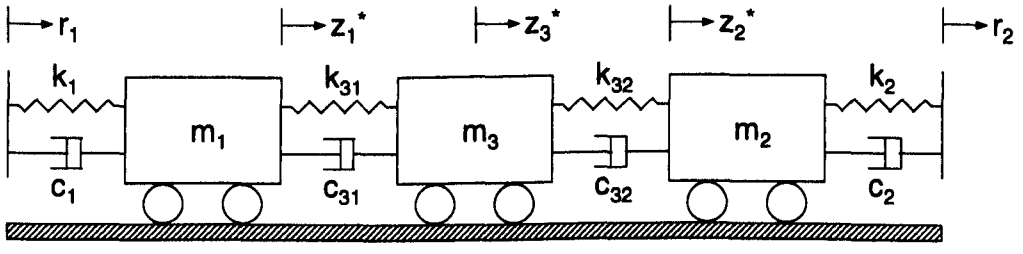


Figure 2.5: Schematic representation of the three mass system

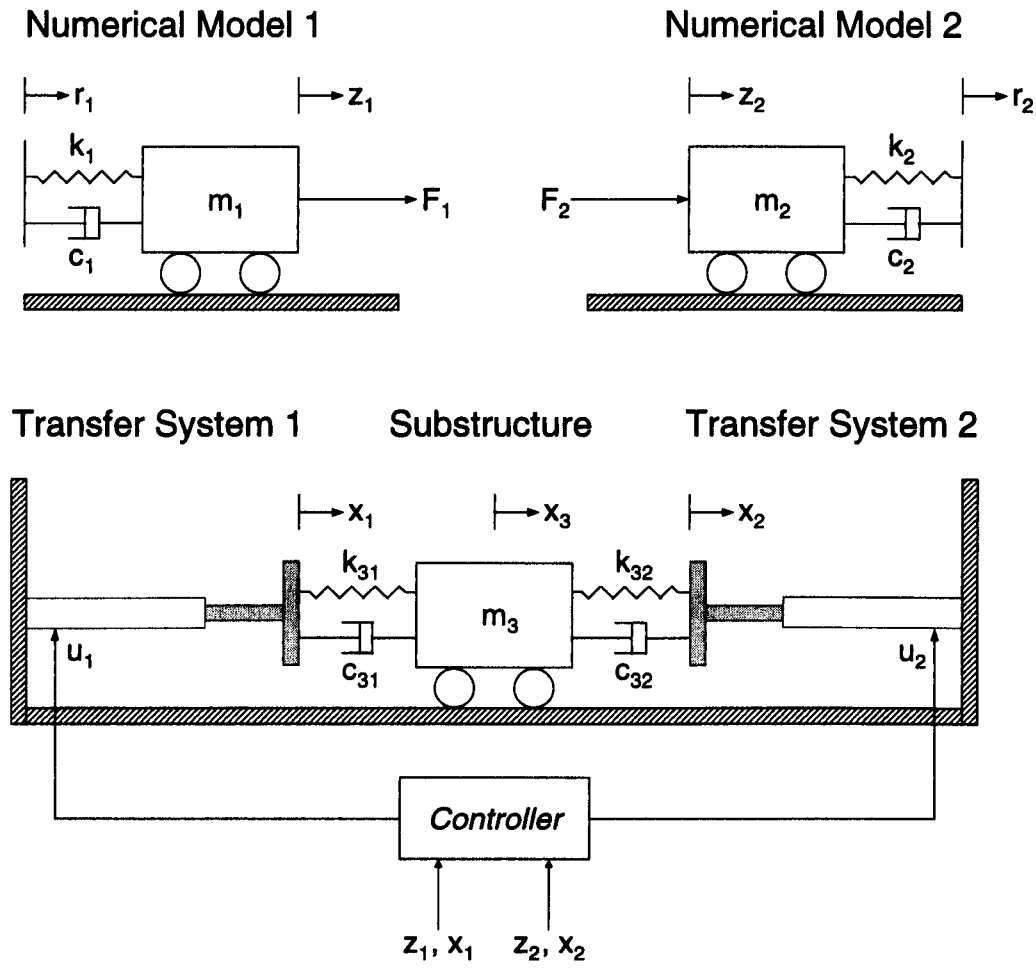


Figure 2.6: Schematic representation of a *substructured* three mass system with two transfer systems

transfer systems via two independent control signals, u_1 and u_2. The influence of the substructure is now represented by two autonomous forces, F_1 and F_2, which are measured experimentally and then imposed back on the numerical models. This new substructured model is shown schematically in Figure 2.6.

From Figure 2.6, the dynamics of the two numerical models can be written as,

$$\begin{aligned} m_1\ddot{z}_1 + c_1(\dot{z}_1 - \dot{r}_1) + k_1(z_1 - r_1) &= F_1, \\ m_2\ddot{z}_2 + c_2(\dot{r}_2 - \dot{z}_2) + k_2(r_2 - z_2) &= F_2. \end{aligned} \tag{2.6}$$

Due to the linear nature of the example considered here, we know explicitly the

forces at each time interval allowing us to measure the accuracy of an individual test, where

$$\begin{aligned} F_1 &= c_{31}(\dot{z}_3 - \dot{z}_1) + k_{31}(z_3 - z_1), \\ F_2 &= c_{32}(\dot{z}_2 - \dot{z}_3) + k_{32}(z_2 - z_3). \end{aligned} \tag{2.7}$$

As in § 2.3.1 we can now use Eq. (2.6) with Eq. (2.7) to assess the performance of the substructuring algorithm and highlight the effect of the delays in the system. This highlights the difficulty in assessing the accuracy of substructuring tests for complex systems, as if Eq. (2.7) was unknown the end results of the test could never be compared with results from the emulated system. This point is discussed in greater depth later in this chapter and in § 4.3.7.

We note that there are now two distinct and independent delay errors. Transfer System 1 (TS1) has the delay τ_1 and correspondingly Transfer System 2 (TS2) has the delay τ_2. Thus, the nature of the critical delay criterion is more complex and is discussed in § 3.2.5. Additionally, as each transfer system has its own individual characteristics it is possible, and in fact highly likely, that the delay magnitudes will not be identical. This can introduce erroneous dynamical effects in the substructure analogous to that found in multiple support excitation experiments [83]. It is therefore beneficial to decouple the dynamics of the transfer systems such that their control can be dealt with independently. This is described in more detail in § 2.4.3.

2.3.3 Experimental set-up

Figure 2.7(a) shows the experimental substructure setup for the multi DOF case study of § 2.3.2. An enlarged view of the substructure with the load cell locations is shown in Figure 2.7(b). This rig can be used for both the single DOF and the multi DOF case studies (as well as many other configurations for a maximum of two actuators) by rearranging the actuators, masses and springs as required.

The actuators are UBA (timing belt and ball screw configuration) linear Servomech actuators with maximum force capacity of 410N and maximum linear speed of

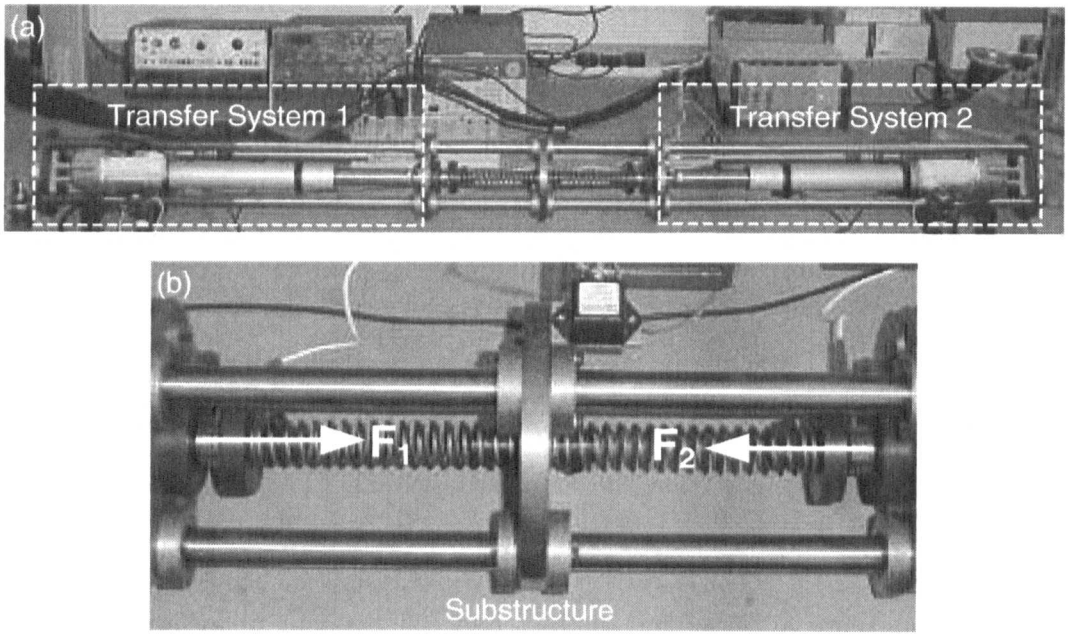


Figure 2.7: (a) Experimental rig set-up of substructured model. (b) Enlarged view of substructure.

640mm/s. As with all actuators there is a drop-off in the maximum force as its driving speed is increased producing a specific performance envelope for linear behaviour. Each actuator is driven independently by a Panasonic Minas Series AC servo motor which is configured only as an analogue amplifier. The servo-mechanical actuators used on the rig can be viewed as an advanced servo system — the AC motors are designed specifically as a high performance motion control device. The motors provide a maximum speed of 5,000 rpm, optional 17-bit absolute encoder resolution, and three different inertia levels for best application suitability. Additionally, different drives are provided to cover a power range of 50 to 5000 watts, and the system may be controlled via Analogue Voltage Signal (speed or torque), or two digital inputs (Step and Direction). For the purposes of our control, we shall use the analogue voltage control as this allows the control algorithm to be fully determined within the substructuring algorithm, rather than additional control being added by the servo unit, whose characteristics can only be viewed as

a "black-box".

Displacements are measured using RDP Electronics DCT captive guided DC LVDT (Linear Variable Differential Transformer) displacement transducers which have a $\pm 0.11\%$ linearity error on full scale deflection of 50mm. Each unit has an internal bearing that guides the armature and built-in dc to dc signal conditioning to help remove noise.

The force applied to the substructure is measured using RDP Electronics model 31 precision miniature tension/compression load cells. The unit is applicable both in tension and compression with linearity $\pm 0.15\%$, hysteresis $\pm 0.15\%$ and non-repeatability $\pm 0.1\%$ of full scale deflection.

Each mass is a constant 2.2kg and connected to the rig via three parallel shafts constraining its motion to one degree-of-freedom with an axial alignment accuracy of ± 0.1mm. Each mass has three LBBP linear ball bearings with double lip seals and raceway plates to reduce friction. Due to the positioning of the load cells, see Figure 2.7(b), the masses attached to the actuators are simply part of the actuator piston as they are rigidly attached. If the load cell was positioned in between the actuator piston and the mass then it will act as *added mass* with its inertia affecting the force signal (the mass of the nylon support can be neglected). In this instance, this must be taken into account in the numerical model, either to compensate for its existence or to alter the substructured system being tested. A range of springs may be attached to achieve differing spring constants, these vary from $k = 1500N/m$ to $k = 9000N/m$, each with a corresponding damping ratio, c, experimentally established through system identification techniques.

2.3.4 Transfer system identification

System identification aims to produce a model of the system based on observed input-output data. There are many ways to describe a system and to estimate

its dynamic properties [84]. No approach will ever lead to a perfect model, however, a good model is often useful since it will directly improve experimentation as uncertainty about the plant dynamics is reduced. Generally, system identification techniques fall into one of two categories, time domain and frequency domain.

Figure 2.8 shows the open loop time domain response of one of the actuators that will be used for experimentation as described in § 2.3.3. Due to the fact that the AC servo motors are being used simply as analogue amplifiers (to power the actuators), the effect is the actuator's response acts like a natural integrator, in effect the actuators are under velocity control — a constant demand voltage will produce a constant speed of the actuator piston, shown by the constant gradient of the *Response* line from approximately 0.51s. Between the time of 0.5s (when the step demand is initiated) and 0.51s there is a transitional period. However, this transition is not smooth, in fact, there is a *deadzone* for the first 4ms of this time frame. All controlled devices have some form of deadzone (which is one of many types of nonlinear phenomena) — it is the time taken for the static friction of the internal mechanism to be overcome such that the active device starts to move. In the case of this actuator, the minimum open loop deadzone time is 4ms, this can be reduced by closing the control loop such that the control demand can compensate

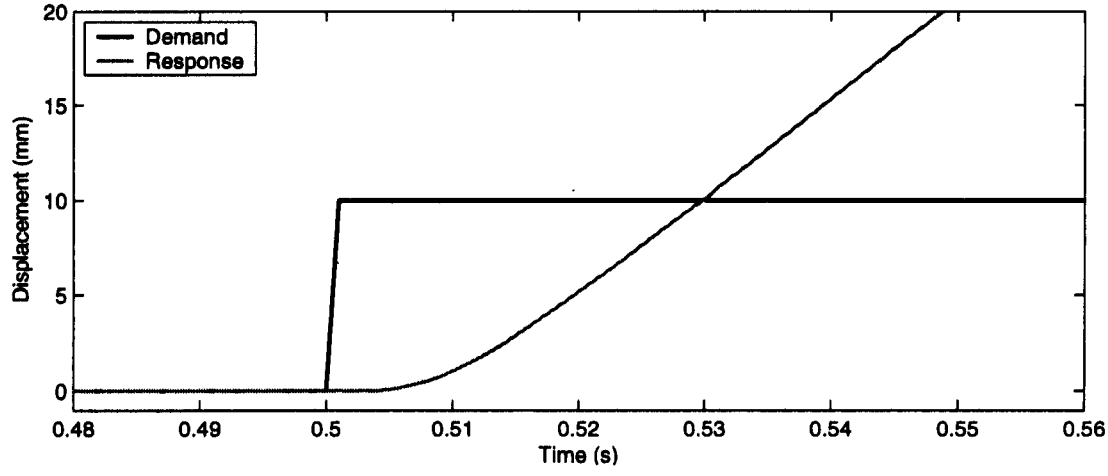


Figure 2.8: Open loop step response for Actuator 1 (Δt = 1ms)

for this nonlinearity.

We now perform an open loop time domain system identification of the same actuator using a swept sine wave signal as an excitation, this will allow us to gain an understanding of the dynamic characteristics of the actuator over the operational range of frequencies. Figure 2.9 shows such a test between 1Hz and 15Hz completed in 60s. As expected, due to the natural integrator characteristics, the actuator overshoots dramatically at low frequency due to the prolonged voltage demand of a given state (+ve or −ve). From these results we can compute an estimated model (transfer function) of the actuator's dynamic properties. For this we have used the *oe*(Output Error) MATLAB function, which attempts to predict the best fitted model from a polynomial based output error algorithm. From Figure 2.9 we extract the open loop transfer function for Actuator 1 to be:

$$G(s) = \frac{y(s)}{u(s)} = \frac{0.7068s + 5700}{s^2 + 62.26s + 31.61}, \tag{2.8}$$

where, $y(s)$ is the output, $u(s)$ is the input and $G(s)$ represents the *plant* [84].

We can now close the control loop and apply the proprietary (linear) controller that will act as the *inner* loop control of the substructuring algorithm — such that the actuator **and** the proprietary controller form the *transfer system*. For simplicity,

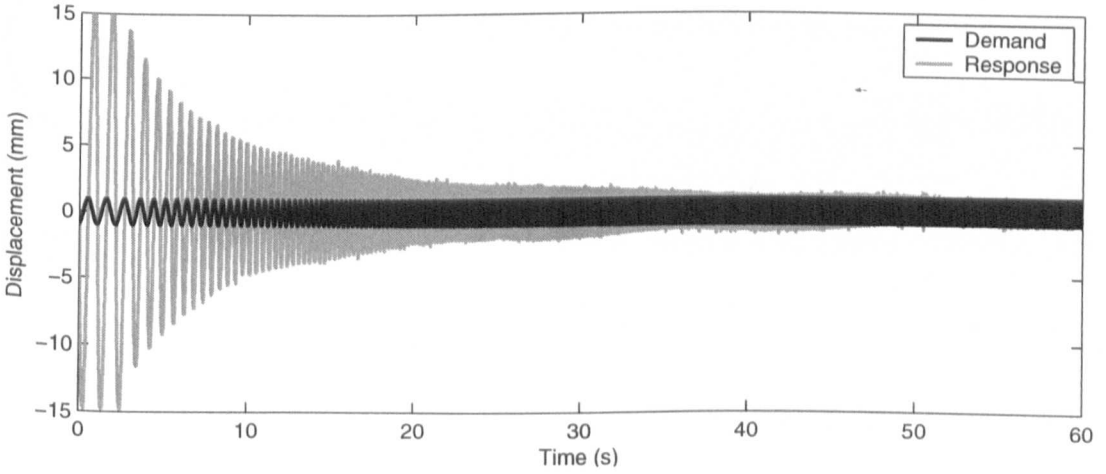


Figure 2.9: Open loop sine sweep response for Actuator 1; 1 - 15 Hz in 60s.

we will only be using a linear proportional (P) controller for the proprietary control rather than a PID (Proportional, Integral and Derivative gains) controller. This is valid as the plant is a stiff system with low steady state error ([84]) which is relatively noisy, therefore any derivative action must be very small. In fact, as will be discussed a number of times in this thesis, we want as predictable a response as possible from the proprietary controller such that there is low uncertainty when applying any *outer* loop control.

The appropriate proportional gain, k_p, must be identified. Figure 2.10 shows the Roots Loci plot for the open loop transfer function, Eq. (2.8), including conversion factor for displacement control in millimeters. We require a fast response from the transfer system to achieve a good representation of the numerical model (which will be in the form of a smooth signal) and to reduce the demands on any outer loop controller. Thus, the closed loop gains should be designed to give a slightly underdamped response $\zeta = 0.7 - 0.8$. As can be seen from Figure 2.10 a loop gain

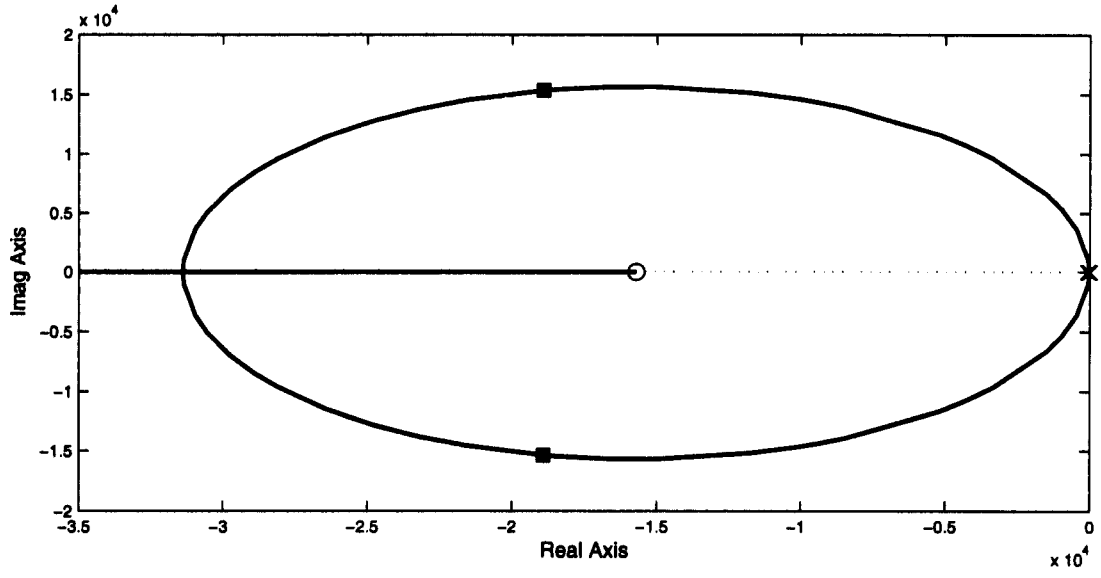
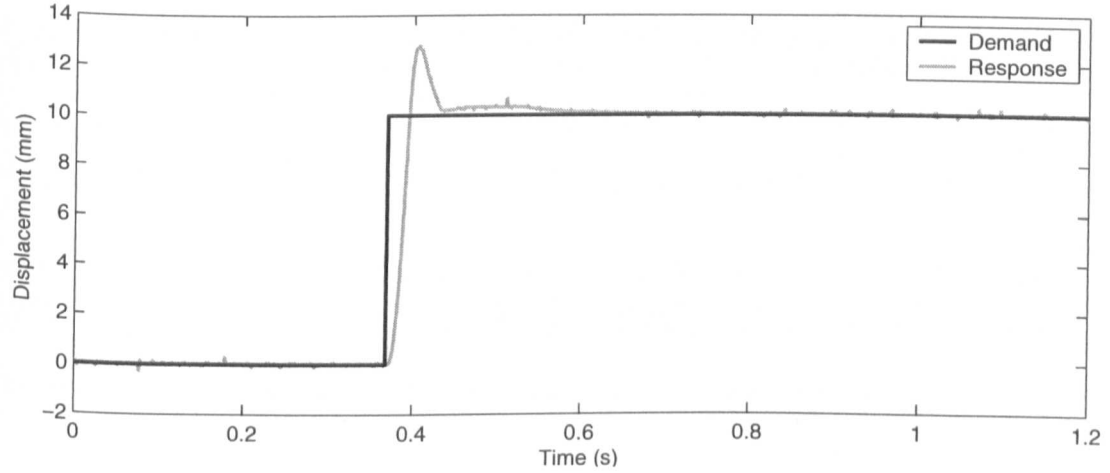


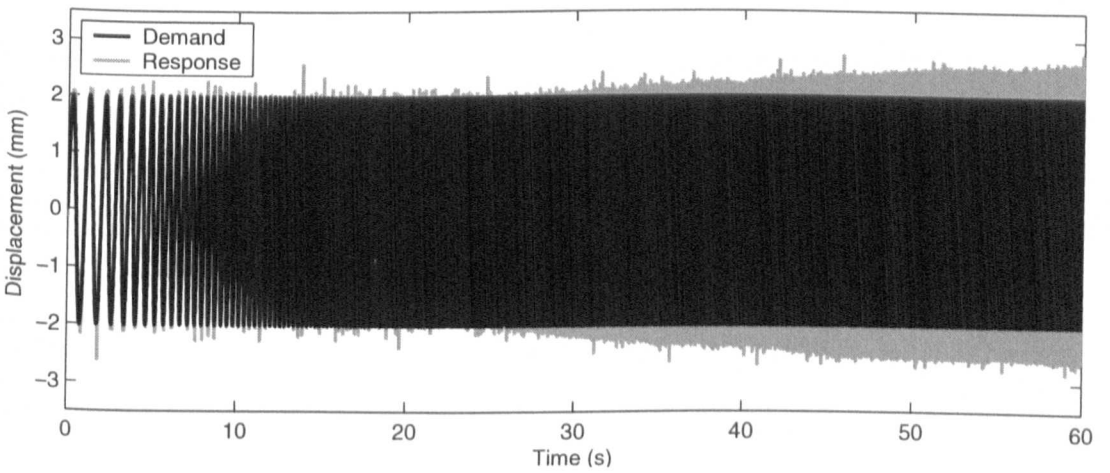
Figure 2.10: Roots Loci for the open loop step response of Transfer System 1, Eq. (2.8). Closed loop poles positioned to give closed loop damping of $\zeta = 0.7 - 0.8$; $\zeta = 0.776$ corresponds to a loop gain of 1.

of $k_p = 1$ gives a damping value of $\zeta = 0.776$. This gives us unity feedback and further simplifies are understanding of the transfer system.

We can now observe the closed loop dynamic response of the transfer system for this proportional gain of $k_p = 1$ in Figure 2.11. Figure 2.11(a) shows the closed loop step response, we observe approximately one overshoot representing a slightly underdamped system as designed in Figure 2.10. Although, it should be noted that the internal mechanisms of the actuator are complex and include a braking device



(a) Step response.



(b) Sine sweep response; 1 - 15 Hz in 60s.

Figure 2.11: Closed loop response for Transfer System 1 with $k_p = 1$.

— the effect of this can be seen at approximately 0.45s changing the response from a classic 2^{nd} order response to being nonlinear. Figure 2.11(b) shows the closed loop response of the transfer system to the sine sweep excitation performed originally for the system identification in Figure 2.9. We observe a gradual increase in overshooting as the frequency increases due to the velocity control characteristics of the AC servo motors. However, more importantly in terms of a substructuring algorithm is the magnitude of the delay. The results from Figure 2.11(b) were run through a post processing algorithm that calculates the delay magnitude at the zero crossing points ($z = 0$) and is shown in Figure 2.12. As the data are sampled at discrete points in time (at a sampling rate of 1kHz) the algorithm can only estimate the delay to the nearest millisecond, however, a distinct trend emerges showing that over this experimental range the response of the actuator can be approximated very well to a system that has a constant delay of approximately 9ms. The trend is slightly less obvious at very low frequency as the response is more significantly effected by noise and the transfer system nonlinearities; we discuss this in more detail in § 5.2. We improve upon the estimation of this delay magnitude when we discuss the concept of delay compensation in Chapter 4.

A similar analysis was performed for the second actuator (required for the multi

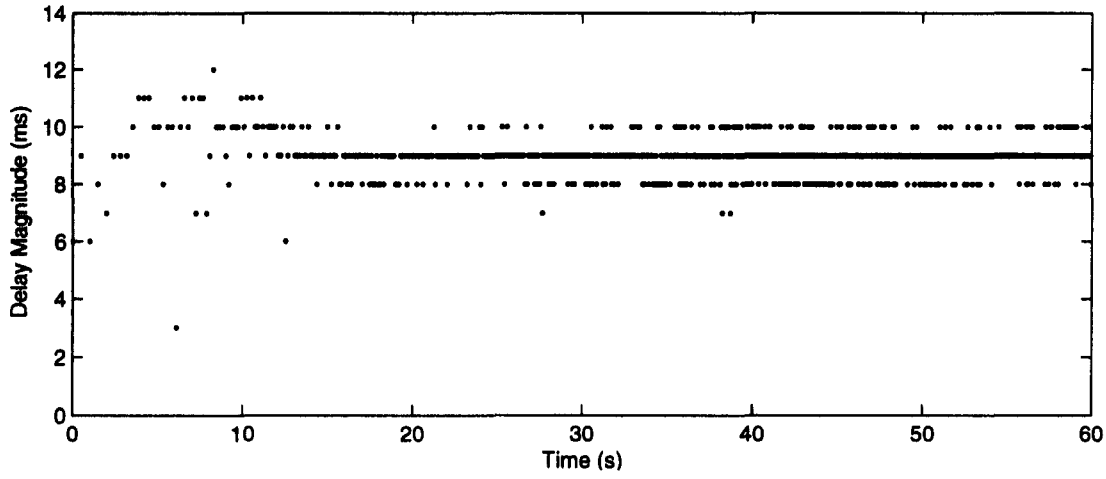


Figure 2.12: Delay magnitude plot for the sine sweep response of Figure 2.11(b).

DOF Substructuring example of § 2.3.2) and gave comparable results (again with a gain value of $k_p = 1$) with an approximate delay of 8ms and similar characteristics, although the exact dynamics are not identical. The fact that the transfer systems are similar is unsurprising as the actuators and servos are of the same type, however, the fact that the dynamics are not identical means that each transfer system must be treated independently in terms of applying an outer loop controller.

2.4 Numerical modelling techniques

An essential part of substructuring is the way in which the remainder of the system (what is left over after the substructure has been removed) is modelled numerically. The accuracy of the numerical modelling techniques (including the stability of the integration strategy used [79, 80]) helps to determine how faithfully the substructuring algorithm can reproduce the dynamics of the original structure. However, the work presented in this thesis is concerned with the development of the experimental side of the substructuring technique, and thus, numerical modelling algorithms do not form a major part of this thesis. The real-time architecture that will be used throughout this work, presented in § 2.4.4, allows the use of well known fixed step size explicit algorithms, such as the 4th order Runge Kutta, for which stability can be assured due to the small sample times which will be employed. Therefore the numerical modelling techniques, described in the remainder of this section, are only used as tools to facilitate the development of our understanding of the experimental side of real-time dynamic substructuring. The study of real-time numerical modelling techniques (such as ultra fast finite element analysis) and the stability of specific mixed operator integration algorithms is therefore beyond the scope of this work but a potential direction of future work as discussed in § 7.2.

In the context of numerical modelling, it is important to comment on the integration schemes used to run these processes in real-time. As real-time substructuring typically has a small sample time, explicit integration schemes can be used as

stability can be assured. It should be noted that this is one of the crucial differences in implementing real-time compared to PsD testing. Due to the time scales being expanded for PsD it is often necessary, or essential, to use an implicit scheme. In fact, mixed implicit-explicit schemes such as the operator splitting approach are most commonly used [47, 57, 80]. To illustrate the differences between implicit and explicit integration we can consider the following linear system

$$\dot{x} = Ax + Bu. \tag{2.9}$$

For the purposes of simulation, this system can be discretised (with sample interval Δt) in two ways

$$1. \quad \dot{x} = \frac{x_{i+1} - x_i}{\Delta t}, \tag{2.10}$$

$$2. \quad \dot{x} = \frac{x_i - x_{i-1}}{\Delta t}. \tag{2.11}$$

Eq. (2.10) gives rise to the forward Euler or *explicit* integration scheme

$$x_{i+1} = x_i + \Delta t[Ax_i + B_i u], \tag{2.12}$$

and Eq. (2.11) gives rise to the backward Euler or *implicit* integration scheme

$$x_i = x_{i-1} + \Delta t[Ax_i + B_i u]. \tag{2.13}$$

For the purposes of implementation, Eq. (2.13) must be rewritten as

$$x_i = [I - \Delta tA]^{-1}[x_{i-1} + \Delta tB_i u]. \tag{2.14}$$

The explicit method gives simple implementation whereas the implicit method requires matrix inversion. However, the explicit method is only stable if

$$\Delta t < \frac{2}{|\lambda|}, \tag{2.15}$$

where λ is the largest eigenvalue of A. If this largest eigenvalue is real, $\lambda = \frac{1}{\varphi}$, where φ is the smallest system time constant. Therefore,

$$\Delta t < 2\varphi. \tag{2.16}$$

If the system is stiff, that is it contains at least one small time constant relative to the dominant time constants, Euler integration is not feasible due to the very small sample interval Δt required. In contrast, the implicit method is stable.

Due to the real-time architecture that is described in § 2.4.4 it is possible to use higher order integration schemes than the Euler method described above. All the experiments presented in this thesis use a 4^{th} order Runge-Kutta scheme. Although, Runge-Kutta in its original form is not real-time compatible (as the 4^{th} step requires information from the following time step) the real-time architecture used includes a modified version which is real-time compatible. The use of a higher order scheme has no effect when comparing the substructure results to the emulated system, as both are similarly affected by the same numerical errors, however, the accuracy of the overall test compared to the real system does increase as the integration scheme produces smaller numerical errors.

The following sections, § 2.4.1 to § 2.4.3, describe the three numerical modelling techniques that will be used at various times throughout the work presented in this thesis.

2.4.1 Transfer function representation

The *transfer function* is the most simple modelling technique and is most commonly used in the field of control engineering. It is a linear estimation of a system $G(s)$ and is defined as the ratio of the Laplace transform[1] of the output variable $y(s)$ to the Laplace transform of the input variable $u(s)$, with all initial conditions assumed to be zero [85].

$$G(s) = \frac{y(s)}{u(s)}. \tag{2.17}$$

A transfer function may be defined only for a linear, stationary and constant parameter system. A nonstationary system, called a time-varying system, has one or

[1]The Laplace transform is a transformation of an arbitrary function $f(t)$ from the time domain into the complex frequency domain $F(s)$ [85].

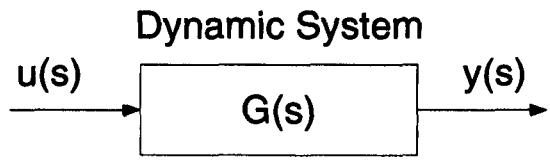


Figure 2.13: A block diagram representation of a general transfer function $G(s)$

more time-varying parameters and therefore the Laplace transformation may not be utilized. Furthermore, a transfer function is an input-output description of the behaviour of the system; thus, the transfer function description does not include any information concerning the internal structure of the system and its behaviour. Eq. (2.17) can be represented by a *block diagram* as shown in Figure 2.13.

In substructuring, we can use this relationship between the output and the input to model the dynamics of the numerical model in place of the general transfer function $G(s)$. Firstly, using the single DOF example of § 2.3.1 we obtain the following 2^{nd} order transfer function for the numerical model of the substructuring algorithm:

$$G(s) = \frac{c.s + k}{m.s^2 + c.s + k} - \frac{F}{m.s^2 + c.s + k}. \tag{2.18}$$

As the dynamics of the substructured system are decoupled, by treating the feedback forces as an autonomous disturbances, the multi DOF example of § 2.3.2 is of the same form as Eq. (2.18) but has two independent models:

$$G_1(s) = \frac{c_1.s + k_1}{m_1.s^2 + c_1.s + k_1} - \frac{F_1}{m_1.s^2 + c_1.s + k_1}, \tag{2.19}$$

$$G_2(s) = \frac{c_2.s + k_2}{m_2.s^2 + c_2.s + k_2} - \frac{F_2}{m_2.s^2 + c_2.s + k_2}. \tag{2.20}$$

Representing the numerical model of a given substructured system in this way is relatively quick and simple, however, it will only provide a linear estimation. Additionally, when the system has various discrete dynamical states (for example the nonlinear piecewise smooth nature of the damper examined in Chapter 6) this method can become cumbersome.

2.4.2 Bond graph representation

A *bond graph* is an explicit graphical tool for capturing the common energy structure of systems and can increase one's insight into the system's behaviour. They provide a concise description of complex systems though vector notation. Moreover, the notation of *causality* provides a tool for the discussion of a system's behaviour, in terms of controllability, observability, fault diagnosis, etc. Through this approach, a physical system can be represented by symbols and lines, identifying the power flow paths. The lumped parameter elements of resistance, capacitance and inertial components are interconnected in an energy conserving way by bonds and junctions resulting in a network structure. Bond graphs are discussed in the text books of Ljung and Glad [86], Cellier [87], Karnopp et al. [88] and the journal special issue of Gawthrop and Scavarda [89] contains an overview of the topic.

Although this thesis is concerned with mechanical systems, a strength of the approach is that it can be uniformly applied across the range of physical domains listed in Table 2.2 as the bond graph approach uses the generic *effort* and *flow* variables to describe a range of physical domains; and thus, we will use "force" in place of "effort" and "velocity" in place of "flow".

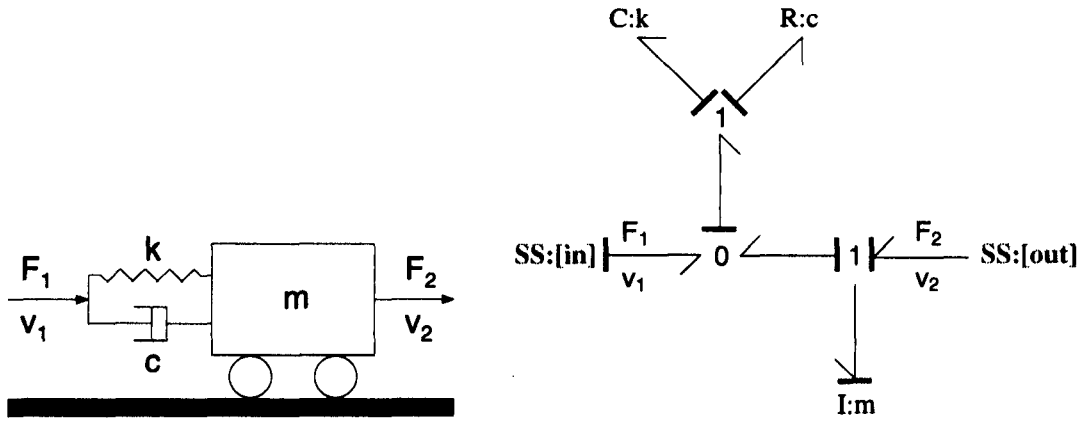
To demonstrate this concept we use the single DOF example of § 2.3.1. Figure 2.14 shows a schematic diagram of the system with its bond graph. The wall excitation is associated with the force–velocity pair F_1–v_1 whereas the right-hand side of the

Domain	Effort	Flow
Mechanical	Force	Velocity
Mechanical	Torque	Ang. velocity
Electrical	Voltage	Current
Hydraulic	Pressure	Flow rate
Thermal	Temperature	Entropy flow

Table 2.2: Physical domains

mass is associated with the force–velocity pair F_2–v_2. For such systems, each bond symbol "⇀" represents such a force–velocity pair; because force × velocity = power, these symbols are also called *power* bonds. The direction of the half-arrow shows the direction regarded as having positive power flow; it thus specifies the sign convention. The **I**, **C** and **R** components of Figure 2.14(b) correspond the the mass, spring and damper components of Figure 2.14(a). The **0** junction is a common *force* junction (all connected bonds have the same force) and the **1** junction is a common *velocity* junction (all connected bonds have the same velocity). The two **SS** components provide connections to other systems, in this case the wall excitation and the substructure.

Assuming that $v_1$and F_2 are imposed on the system (inputs) and that F_1 and v_2 are the corresponding outputs, the system can be written as a transfer function as shown in § 2.4.1. However, the bond graph approach does not insist on this input/output choice; the choice of input and output is associated with the bond graph concept of *causality* [86–88]. Because the concept of causality is so important, the bond graph approach has a special notation: the *causal stroke*. With reference to Figure 2.14(b), causality is indicated by a line perpendicular to a bond at the end of the bond where effort is imposed; flow is imposed at the other end. If the



(a) Schematic (b) Bond Graph

Figure 2.14: Single DOF case study from § 2.3.1

causal stroke on the mass, **I**, were reversed, this would correspond to an improper transfer function representation, in effect, inverting gravity.

However, one benefit of using bond graphs is that it is not only the numerical model that can be incorporated. The technique allows us to incorporate other ideas into the same framework, such as the lag compensation technique described in § 4.4.1.

2.4.3 State-space representation using S-functions

The state-space description provides the dynamics of a system as a set of coupled differential equations with internal variables known as *state variables*, together with a set of algebraic equations that combine the state variables into physical output variables. The concept of the *state* of a dynamic system refers to a minimum set of variables, that fully describe the system and its response to any given set of inputs [85]. A mathematical description of the system in terms of a minimum set of variables $\phi_i(t)$, where $i = 1, \ldots, n$, together with knowledge of those variables at an initial time t_0 and the system inputs for time $t \geq t_0$, are sufficient to predict the future system state and outputs for all time $t > t_0$. This definition asserts that the dynamic behaviour of a state-determined system is completely characterized by the response of the set of n variables $\phi_i(t)$, where the number n is defined to be the order of the system.

Large classes of engineering, biological, social and economic systems may be represented by state-determined system models. System models constructed with linear one-port elements (such as the mass-spring-damper elements of the two case studies of § 2.3) are state-determined system models. For such systems the number of state variables, n, is equal to the number of independent energy storage elements in the system. The values of the state variables at any time t specify the energy of each energy storage element within the system and therefore the total system energy, and the time derivatives of the state variables determine the rate of change of the system energy. Furthermore, the values of the system state variables at any time t provide

sufficient information to determine the values of all other variables in the system at that time.

In the general case the form of the n state equations is:

$$\begin{aligned} \dot{\phi}_1 &= f_1(\boldsymbol{\phi}, \mathbf{r}, t), \\ \dot{\phi}_2 &= f_2(\boldsymbol{\phi}, \mathbf{r}, t), \\ \vdots &= \vdots \\ \dot{\phi}_n &= f_n(\boldsymbol{\phi}, \mathbf{r}, t), \end{aligned} \tag{2.21}$$

where, $r_1(t), \ldots, r_p(t)$ are the system inputs, $\phi_1(t), \ldots, \phi_n(t)$ are the state variables and $\dot{\phi}_i = d\phi_i/dt$. Each of the functions $f_i(\boldsymbol{\phi}, \mathbf{r}, t), (i = 1, \ldots, n)$ may be a general nonlinear, time varying function of the state variables, the system inputs, and time. It is common to express the state equations in a vector form, thus the set of n equations in Eq. (2.21) may be written as $\dot{\boldsymbol{\phi}} = \mathbf{f}(\boldsymbol{\phi}, \mathbf{r}, t)$.

We now restrict the a description of the system to be linear and time-invariant (a system that can be described by linear differential equations with constant coefficients). For such a system of order n, and with p inputs, Eq. (2.21) becomes a set of n coupled first-order linear differential equations with constant coefficients:

$$\begin{aligned} \dot{\phi}_1 &= a_{11}\phi_1 + a_{12}\phi_2 + \ldots + a_{1n}\phi_n + b_{11}r_1 + \ldots + b_{1p}r_p, \\ \dot{\phi}_2 &= a_{21}\phi_1 + a_{22}\phi_2 + \ldots + a_{2n}\phi_n + b_{21}r_1 + \ldots + b_{2p}r_p, \\ \vdots &= \vdots \\ \dot{\phi}_n &= a_{n1}\phi_1 + a_{n2}\phi_2 + \ldots + a_{nn}\phi_n + b_{n1}r_1 + \ldots + b_{np}r_p, \end{aligned} \tag{2.22}$$

where the coefficients a_{ij} and b_{ij} are constants that describe the system. Eq. (2.25) may be written compactly in a matrix form:

$$\frac{d}{dt}\begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_n \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{n1} \\ a_{21} & a_{22} & \ldots & a_{n2} \\ \vdots & & & \vdots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_n \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} & \ldots & b_{1n} \\ b_{21} & b_{22} & \ldots & b_{2n} \\ \vdots & & & \vdots \\ b_{n1} & b_{n2} & \ldots & b_{np} \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_p \end{bmatrix}, \tag{2.23}$$

which can be written in vector format as

$$\dot{\boldsymbol{\phi}} = \boldsymbol{A}\boldsymbol{\phi} + \boldsymbol{B}\boldsymbol{r}, \tag{2.24}$$

where the state vector ϕ is a column vector of length n, the input vector $\mathbf{r}$ is a column vector of length p, $\mathbf{A}$ is an n x n square matrix of the constant coefficients a_{ij}, and $\mathbf{B}$ is an n x r matrix of the coefficients b_{ij} that weight the inputs.

For the case of substructuring, there is an additional autonomous input from the force fed back form the substructure. For the single DOF case study of § 2.3.1 we can rewrite Eq. (3.1) in state-space format as

$$\begin{aligned} \phi_1 &= z \\ \phi_2 &= \dot{z} \end{aligned} \tag{2.25}$$

$$\frac{d}{dt}\begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix}\begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix}\begin{bmatrix} r \\ \dot{r} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{F}{m} \end{bmatrix}. \tag{2.26}$$

Thus, Eq. (2.24) is extended to

$$\dot{\boldsymbol{\phi}} = \boldsymbol{A\phi} + \boldsymbol{Br} + \boldsymbol{C}, \tag{2.27}$$

where $\mathbf{C}$ represents the feedback force vector of $\mathbf{C} = [0 \;\; F/m]^T$ from the substructure. When considering the multi DOF system of § 2.3.2, the number of states increases to $n = 4$, where, $\phi_{1,2,3,4} = [z_1 \;\; \dot{z}_1 \;\; z_2 \;\; \dot{z}_2]^T$. Converting the numerical model equations of Eq. (2.6) into state space form, the constant coefficient matrices are given by:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{k_1}{m_1} & -\frac{c_1}{m_1} & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -\frac{k_2}{m_2} & -\frac{c_2}{m_2} \end{bmatrix}, \tag{2.28}$$

$$\mathbf{B} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ -\frac{k_1}{m_1} & -\frac{c_1}{m_1} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{k_2}{m_2} & -\frac{c_2}{m_2} \end{bmatrix}, \tag{2.29}$$

$$\mathbf{C} = \begin{bmatrix} 0 \\ \frac{F_1}{m_1} \\ 0 \\ \frac{F_2}{m_2} \end{bmatrix}. \tag{2.30}$$

Note, how treating the feedback forces, F_1 and F_2, as autonomous disturbances the two numerical models are completely decoupled. Therefore, as the numerical model outputs act as the *demand* signals for transfer systems, the control of the transfer systems is also decoupled.

In order to implement the numerical model in state-space representation in the context of the real-time architecture (described in § 2.4.4) an *S-Function* must be used (in the MATLAB environment). S-Functions are stand-alone C modules, encoded in a strict function based manner such that they can be compiled into stand alone dynamic link library (*.dll file) and implemented directly into the real-time digital signal processor (DSP) during the build and compile phase. The real-time model data structure encapsulates model data and associated information necessary to fully describe the model. Refer to Appendix A for an example S-Function of the multi-DOF case study of § 2.3.2. An additional, but significant, benefit of encoding the numerical model in a state-space format is that supplementary code can be embedded into the S-Function. This allows the numerical model to be significantly more complicated, allowing for nonlinear and conditional behaviour to be specifically integrated into the model itself, such as found in more complex systems as described in Chapter 6, Appendix B.

2.4.4 Real-time programming

When people talk about real-time, they generally mean "right away" or "fast". A standard programme, such as Microsoft Word, that a general computer user might use, or a text editor that a system programmer might use needs to be fast and responsive, but if it is delayed now and then it is not that important. Generally,

we just want things to happen fast, in development terms this is usually described as *low-latency*. For general-purpose computer systems, "fast" translates to average case performance. However, fast does not imply *real-time*.

A real-time system is one that has deadlines that cannot be missed. For example, consider the control of a robot arm that lifts partially assembled automobiles from one assembly station to another. In order to position the arm correctly, the computer must monitor its movement and stop it precisely 5.2ms after it starts. These timing constraints make this a hard real-time system, where average case performance will not do, stopping the arm 7.1ms after it starts one time and 3.4ms after it starts the next is just not acceptable. Even software that should usually meet timing deadlines, such as video drivers, can afford a hitch now and then. A missed video frame will not cause the damage of a missed robot arm control message. In real-time systems literature (for example, Buttazzo [90]) the text editor is considered to be *non* real-time and the video display would be called *soft* real-time. Only the robot controller would be called a *hard* real-time system. The distinctions are as follows:

Hard real-time: An operating system is considered to be hard real-time if all time constraints imposed by the external world, so-called deadlines, are strictly met within a predefined tolerance, both for a priori deadlines which can be scheduled, and for sporadic deadlines such as interrupts, i.e. the worst case must be within the tolerance, e.g. a real-time process is scheduled within a tolerance of 1ms any time when it ought to and the interrupt response time for any interrupt issued by a pre-selected device is less than 100μs.

Soft real-time: An operating system is considered to be soft real-time if all time constraints imposed by the external world, so-called deadlines, are met in a statistical sense, i.e. the mean value of schedule time deviation is less than a predefined tolerance, e.g. a process is scheduled within a mean tolerance of 1ms, but it may happen, that sometimes the scheduling delay is bigger.

In the 1980's, hard real-time applications were simple enough to be controlled by

dedicated, custom, isolated hardware. However, modern real-time applications must control highly complex systems which are far more general and diverse in purpose. Furthermore, as demands for speed and quality of service increase, applications that have never required it before have begun to require hard real-time support. A CD player that makes a popping sound once in a while is okay for casual listening but is not acceptable for a professional music editing system. The problem is that to deliver the tight worst-case timing needed for hard real-time, operating systems need to be simple, small and predictable. But delivering the sophisticated services that modern applications need is beyond the capabilities of simple, small, predictable operating systems. When you try to put real-time inside a general-purpose operating system, or try to put complex services in a small real-time operating system, you end up with something that does neither task well and where non real-time services can interfere with the execution of real-time services.

Therefore, to a implement real-time dynamic substructuring experimentally we have used a DSP (Digital Signal Processing) processor, specifically the dSpace DS1104 R&D Controller Board running on hardware architecture of MPC8240 (PowerPC 603e core) at 250 MHz with 32 MB synchronous DRAM. This allows the "best of both worlds"; all the development of the substructuring algorithm can be done on an existing non real-time PC with all the benefits of a general-purpose computer system and all the hard real-time applications handled separately on the DSP processor. Explicitly, the substructuring algorithm can be designed using the block diagram-based modelling tool MATLAB/Simulink (which is fully integrated into dSpace architecture) and then compiled and built into the dedicated DSP hardware once complete. During the compiling phase the substructuring algorithm is stringently validated for programming violations before being built into the DSP processor where it is checked for any hard real-time violations. The dSpace companion software ControlDesk is used for online analysis, providing soft real-time access to the hard real-time application on the general-purpose computer. In this way the developer has all the advantages of using a standard PC but can ensure hard real-time constraints.

2.5 Synchronisation theory

Synchronisation *subspace plots* are used to show the effectiveness of the control algorithm, the *local* error, by plotting the *actual* verses the *desired* responses in a 2D vector space, [91]. A subspace plot shows the amplitude accuracy and the magnitude of delay of the transfer system coupled together at any one time. Perfect synchronisation is represented by a diagonal straight line with maxima and minima of the reference signal. Any reduction in synchronisation is seen as a deviation from this idealized line. For periodic wall excitation conditions these plots build up into a repeating pattern, which can appear complex. However, the individual components of amplitude and delay produce their own specific and identifiable patterns if evaluated separately.

The result of varying the amplitude accuracy is to change the angular orientation of the subspace plot compared to the idealized line. Figure 2.15 shows the result of increasing amplitude accuracy from a 0% to 25%, 50% and then 75% of perfect synchronisation. Continuing this trend, the angular orientation further increases to a limit of a 90° (vertical) line which is the outcome of an infinite plant response.

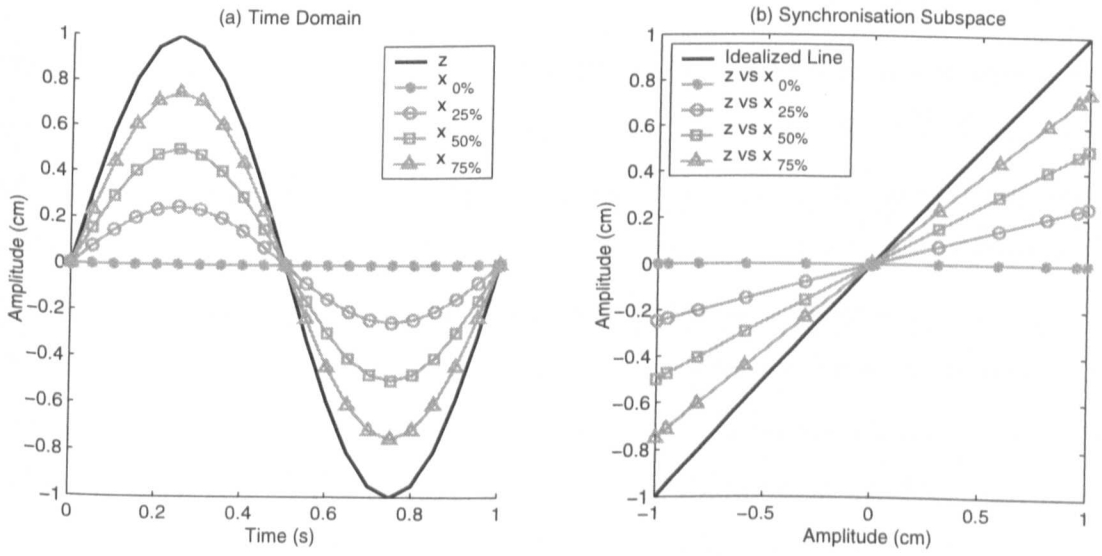


Figure 2.15: Effect of increasing amplitude accuracy with zero delay.

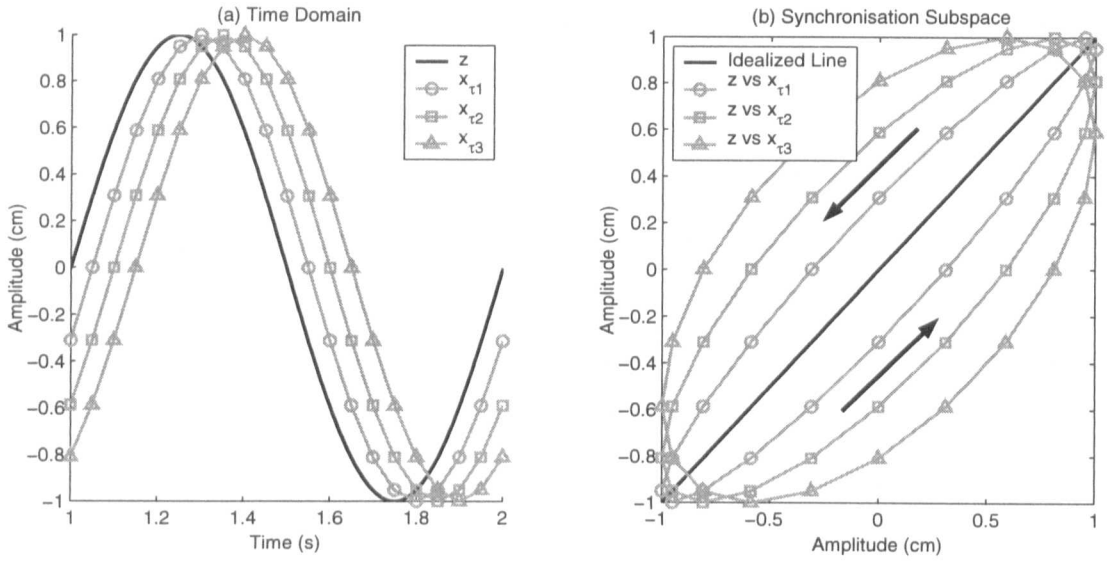


Figure 2.16: Effect of increasing delay with perfect amplitude accuracy.

The consequence of introducing a constant delay between the reference signal and the plant response is to transform the idealized straight line into an ellipse (anti-clockwise implies negative damping, clockwise implies positive damping) as shown by Figure 2.16(b). The greater the delay, the larger the width of the minor axis of the ellipse, with the change being proportional to the delay magnitude. If the delay is not constant through one period, then the ellipse no longer has a uniform shape.

In a typical subspace plot however, the effect of amplitude and delay are coupled together with both being able to vary independently through a single period and thus producing more complicated patterns. Therefore, a typical subspace plot for a transfer system following a given reference signal under proprietary control is more closely related to that shown in Figure 2.17. The linear proprietary controller is designed to reduce uncertainty such that we can achieve a repeatable dynamic response from the actuator. Here, we can see a high amplitude accuracy with a relatively constant delay which increases for a short time after each peak is reached (the actuator deadzone). For multi transfer system substructuring (when significant cross-coupling between the numerical models can be generated) or for more complex excitation conditions the reference would no longer be a single sinusoid. Figure 2.18

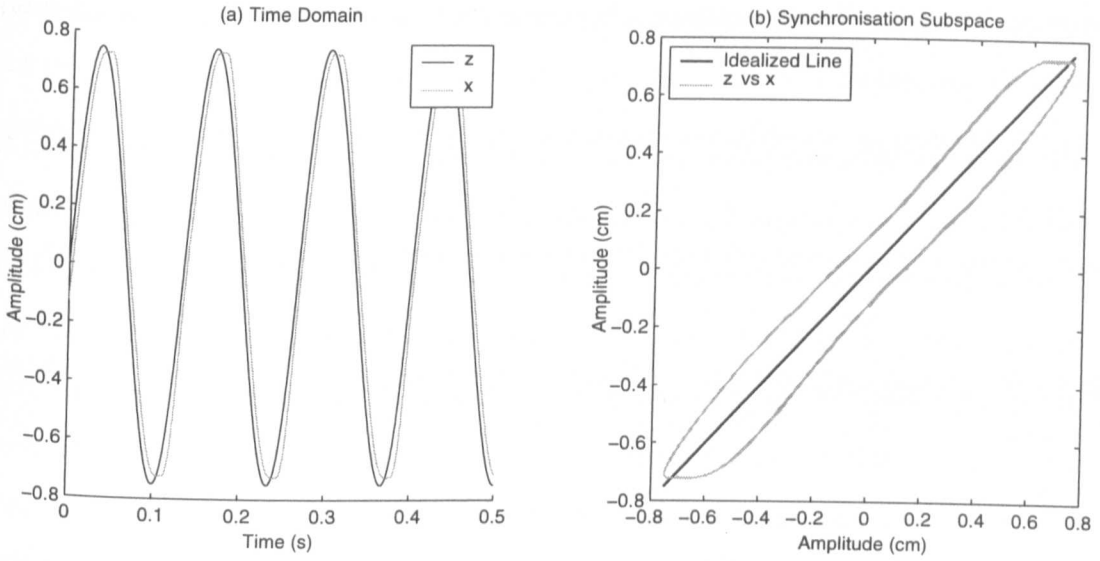


Figure 2.17: A typical subspace plot for a single actuator being sinusoidally controlled using a P controller.

shows a typical subspace plot for a compound sinusoid when the transfer system is under high demand. The compound sinusoid reference results in the idealized diagonal line changing in length according to the maxima and minima of the current

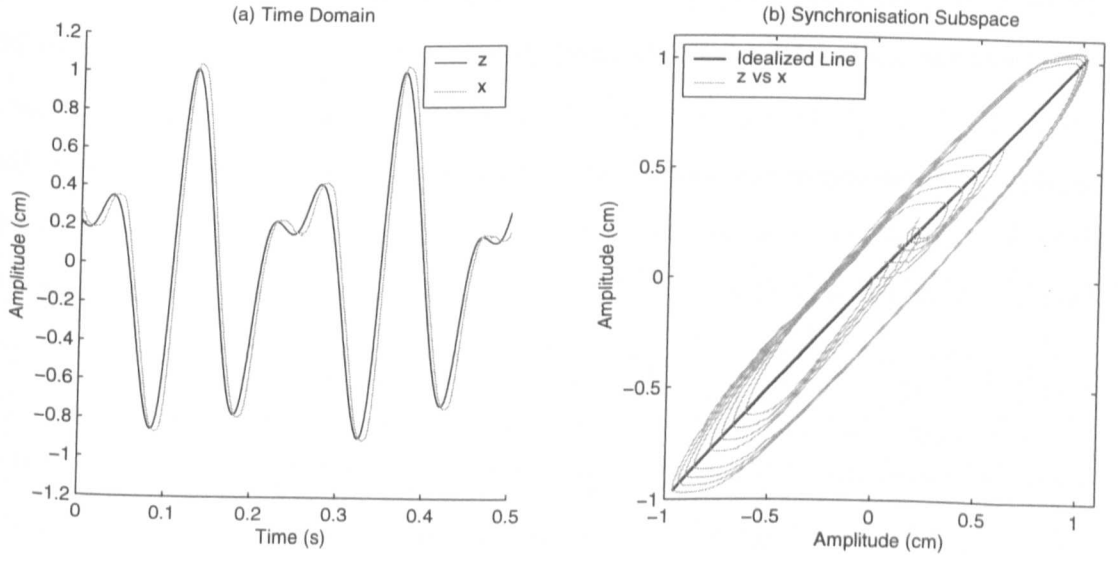


Figure 2.18: A typical subspace plot for multi transfer system substructure test controlled using a P controller.

part of the signal. The magnitude of both accuracy measures now varies considerably during each period.

Although the subspace plots become increasingly complex as the system becomes realistic, the visual interpretation remains constant. Simple linear controllers can be tuned online without performing any further system identification and the adaption characteristics of adaptive controllers can be viewed as they happen by observing the specific shape of the subspace plot. Additionally, this gives the user an initial and instant guide to the accuracy of an individual test and where the potential sources of errors may lie before any detailed post process procedures are carried out. This can be extremely useful in gaining a greater understanding of the experimental characteristics and limitations along with highlighting the major contributing factors that limit the level of synchronisation.

2.6 Effect of delay errors in the substructuring algorithm

There is an important difference between the difficulties faced in a standard control problem to that faced when performing substructuring. For substructuring, the reference signal (i.e. the control demand) for each transfer system is not known at the start of each time step as in a normal control problem, but must be created in its respective numerical model during each time period. A small delay in the transfer system response introduces a corresponding error in the feedback force vector, as shown described in § 2.3.1 and § 2.3.2, which can be thought of as adding negative damping to the system [61]. This discrepancy has the effect of first reducing the accuracy of the numerical models (compared to the emulated system) until the magnitude of the synchronisation delay increases to such a degree that a sign change for the damping of the overall system occurs. At this point instability of the substructuring algorithm is observed and is characterized by the onset of oscillations with exponential growth [69].

Therefore, the aim of the control algorithm is to achieve synchronisation between the

desired interface displacement of the numerical models, z, and the *actual* position of the transfer systems, x. However, under just the *inner-loop* linear control of the proprietary controller, a transfer system will always be subject to some form of delay, τ, which can either be characterized as a pure delay or as a frequency dependent delay (lag) depending on the type of actuator. In fact the nature of this delay error in the substructuring algorithm can be represented by two coupled components (as introduced § 2.1). For the single DOF example of § 2.3.1 we can express the error as

$$e = e_1(z^*, z, t) \; + \; e_2(z, x, t), \tag{2.31}$$

where, e_1 is a function which describes the accuracy of the numerical models compared to the appropriate variable in the complete emulated system (the **numerical model error**):

$$e_1 = (z^* - z), \tag{2.32}$$

and e_2 represents the degree of synchronisation between each transfer system and its numerical model (the **local control error**):

$$e_2 = (z - x). \tag{2.33}$$

Therefore, by combining e_1 and e_2 we can get a **global** measure of the accuracy of the substructuring test which relates the emulated system coordinates z^* to the actual displacement of the transfer systems x. However, when substructuring complex systems it is not possible to compute z^*, and the only measure of accuracy is the degree of synchronisation e_2.

As previously stated, the numerical model coordinates z are, in effect, a delayed function of the transfer system x (as the force vector F will be subjected to the same delay τ as the transfer system), such that

$$z = f(r(t), x(t - \tau)), \tag{2.34}$$

This highlights the nature of the coupling between e_1 and e_2. In the ideal case, achieving perfect synchronisation by removing the delay τ from the transfer systems

will result in $e_2 \rightarrow 0$. This in turn will mean that the correct force vector $F(t)$ will be added into the numerical models at the correct time, such that $e_1 \rightarrow 0$ and the substructured system will replicate the dynamics of the emulated system. This argument allows us to propose the following:

Proposition 1 *If the synchronisation error, $e_2 = 0$, for all time $t \geq 0$ during a substructuring test then $x = z$ and $e_1 = 0$ such that the substructured model exactly replicates the dynamics of the emulated system.*

However, in practice the synchronisation error, e_2, can never be exactly equal to zero in a real substructuring test so the practical interpretation of Proposition 1 is that as $e_2 \rightarrow 0$ the substructured model more closely replicates the dynamics of the emulated system. The significance of Proposition 1 is that it gives an indication of the accuracy of a substructured system using the only measurable quantity of error e_2, the local control error.

We can see the practical implications of Proposition 1 by observing the results from numerical simulations of the single DOF substructured system. Figure 2.19 shows three such tests with varying amounts of modelled transfer system delay. Figure 2.19 (a) shows the case for zero synchronisation error, as can be seen from the subspace plot in Figure 2.19 (a2). As expected, as there is zero numerical model error ($e_1 = 0$ as $e_2 = 0$) the substructured dynamics exactly matches that of the emulated system. As the modelled transfer system delay is increased ($e_2 \neq 0$) we see a resulting error in the numerical model ($e_1 \neq 0$) as shown in Figure 2.19 (b) and (c). As can be seen from Figure 2.19 (b2) and (c2), there is a consistent level of synchronisation throughout the entirety of each simulation. However, the dynamical response of the numerical model in Figure 2.19 (b1) and (c1) are significantly different. When the modelled delay is 6ms (Figure 2.19 (b1)), we observe a stable numerical model with a bounded error, however, when the delay is increased to 7ms (Figure 2.19 (c1)) exponentially growing oscillations are observed. Thus, we can infer that the critical delay, τ_c, occurs at some point between these two delay magnitudes for

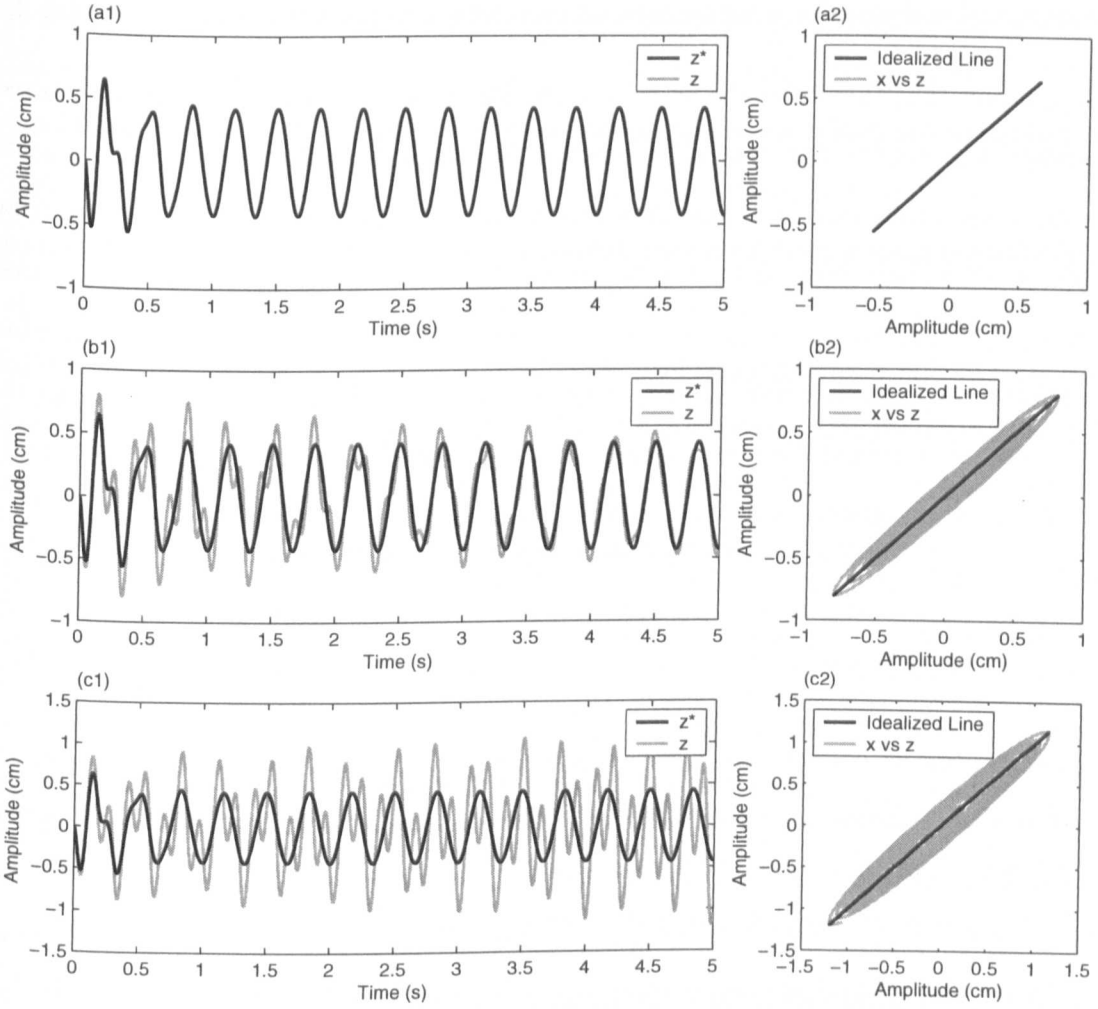


Figure 2.19: Effect of delay on a substructuring algorithm; numerical simulation of the single DOF substructured system from § 2.3.1; System parameters: $m = 2.2$kg, $k = k_s = 2250$N/m, $c = 15$Ns/m. Magnitude of modelled (pure) delay, Panel: (a), $\tau = 0$ms; (b), $\tau = 6$ms; (c), $\tau = 7$ms.

this substructured system with the system parameters as given in the caption for Figure 2.19. From an accuracy standpoint alone it is clear that the primary control objective should be to minimize e_2, however the size of the delay τ also has a significant effect on the stability of the substructuring algorithm as a whole. Any error in e_2 will result in a corresponding error in e_1 and thus propagate to the next time step leading to potential instability of the substructuring algorithm. The

concept of stability for a substructured system is discussed in detail in Chapter 3.

It is a fact that the stiffer the system or the lower the damping in the substructured system the harder the controller must work to retain stability. For this reason, robustness is an essential consideration for this type of testing. In Chapter 5 we apply tools from control theory to study the robust stability of the simple case study systems. In particular we focus on the effect of unmodelled delays and other uncertainties which occur during substructuring. This allows us to develop the concept of a *robust transfer system design methodology* that can be consistently applied to any generic substructuring system.

2.7 Conclusion

In this thesis we consider the hybrid experimental-numerical testing technique known as *real-time dynamic substructuring.* The technique involves creating a hybrid model of the whole system by combining an experimental test piece with one or more numerical models describing the remainder of the system. The virtual testing environment produced mimics the dynamic characteristics of the complete system if done correctly. As only part of the structure is experimentally tested it allows engineers to view the behaviour of critical elements under dynamic loading at full scale. In this chapter we have introduced a number of critical areas which need to be carefully designed into a comprehensive testing strategy in order to achieve successful testing. Chapters 3 to 5 discuss these in depth using the small scale case studies of § 2.3.1 and § 2.3.2 to demonstrate the fundamental principles behind the experimental side of substructuring. So far the technique has been developed successfully using expanded time scales, known as PsD. As testing is carried out quasi-statically, any time-dependent behaviour of the test specimen is lost. Implementing the substructuring process in real-time means that the damping and inertial components of the substructure dynamics are retained.

The issue for substructuring is that as the complete system is split up into its

constituent parts, a controlled system must be employed to exert the influence of the numerical model(s) on the substructure. This influence is obtained through the use of transfer system(s) acting directly on the substructure, which are controlled to follow the appropriate output from each respective numerical model. Delays arise naturally as it is not possible for any controlled system to react instantaneously to a change of state as prescribed by the numerical model, resulting in the overall delay of the transfer system, τ. In a standard control problem this would not be a critical issue, only one of accuracy, but as the force(s) between the transfer system(s) and the substructure are fed back into the numerical model(s) at the same time this gives a form of bi-directional coupling. In some situations the transfer system delay may be so small as to be negligible, but the typical situation in substructuring is that this delay is large enough to have a significant influence on the overall dynamics of the substructured system. The stability criterion, the maximum (or critical) transfer system delay, τ_c, for a substructured system is discussed in detail in Chapter 3.

It is therefore an essential condition that the transfer system delay is less than the critical delay in order to ensure that the substructured algorithm is stable. In Chapter 4, we present two strategies for compensating for the transfer system error, by either characterising it as a pure delay or as a lag (dependent on the frequency of excitation). This highlights how the experimental side of substructuring can be viewed as a control problem, although unconventional.

For complex substructured systems, or those that have a small critical delay, the harder the outer-loop controller must work to retain stability. For this reason, robustness is an essential consideration for this type of testing. In Chapter 5 we apply tools from control theory to better understand the robust stability of a substructured system which allows us to develop the a methodology that can help achieve successful testing for any generic substructuring system.

The knowledge gained from these experiments is then applied to an industrial example of substructuring in Chapter 6, where it would be impossible to understand

the direct influence of the substructure on the entire system in any other way.

Chapter 3

Stability criteria for a substructured system

SUMMARY: This chapter presents the concept of stability in terms of a real-time substructuring algorithm. The delay errors (due to the inner-loop control of the transfer systems) are firstly shown to reduce the dynamical accuracy of the numerical model(s) before instability is observed at the point where the critical limit is reached, characterized by the onset of oscillations with positive exponential growth. We present two approaches to calculating the critical limit of stability for a given system.

3.1 Introduction

The focus of this chapter is on the principle of stability for a given substructured system. The aim is to develop an understanding of the effect of *delay errors* that are always present in a substructuring algorithm. This has been briefly discussed in § 2.6 in general terms, here we express a formal argument about the stability criteria for any given substructured system.

Delays arise naturally, because it is not possible for any controlled plant to react

instantaneously to a change of state as prescribed by the numerical model. There are a number of different delays which combine together to give the overall delay of the transfer system (including data acquisition, computation, digital signal processing and the actuator delay itself) which are combined into one overall synchronisation error, the *local* error e_2 as described in § 2.6. This synchronisation error describes both the delay magnitude and the amplitude error of the closed loop response of the actuator (resulting from the inner-loop proprietary controller). In some situations the transfer system synchronisation error may be so small as to be negligible, but the typical situation in substructuring is that the delay is large enough to have a significant influence on the overall dynamics of the substructured system. The influence is due to the fact that the experimental force(s), that are fed back from the substructure into the numerical model(s), are also delayed by the same magnitude as the transfer system(s) (assuming no filtering of the measured signal) and change the control demand for the next time step. An important criterion for this type of delayed system is its bounds of stability which are defined by the system parameters and the delays within the system. The limit of stability is defined by the critical delay, τ_c. As stated in § 2.1, if the response delay of the transfer system, τ, is of such a magnitude that it goes past this critical limit, τ_c, then the numerical model will become unstable and is characterized by a function of positive exponential growth. This is called the delayed critical limit and is the most significant in terms of the substructuring algorithm. However, as shown in § 2.1 Table 2.1, there is also a forward critical limit, τ_f, which only becomes important in context of the delay compensation scheme employed and is discussed in Chapter 5.

A number of techniques have been proposed to assess the stability of a substructured system. Horiuchi et al. [61] used an energy analysis of periodic orbits to equate the time delay to a form of negative damping with instability occurring at the point of sign change for the damping of the overall system. However, a limitation of this approach was that the form of the transfer system displacement x and the feedback force F needed to approximated to harmonic functions. Lim et al. [72] examined

how the poles of a substructured system, and therefore its stability, are effected by changing the magnitudes of a fixed gain version of an adaptive controller in continuous time. In a similar vein, Darby et al. [73] studied the position of the discrete closed-loop poles of a multi DOF system under the influence of two variable delays. Stability is determined by whether or not the poles lay within the unit circle. Wallace et al. [69] employed the method of modelling the substructured system with *delay differential equations* (DDEs), which are derived from the ODE model of the system by explicitly including the delay(s) due to the transfer system(s). The advantage of DDE modelling is that powerful analytical and numerical methods can be used to determine the stability of the DDE model and, hence, of the substructured system. This technique is discussed in depth in § 3.2. Finally, Gawthrop et al. [74] applied tools from control theory to study the robust stability of a generic linear substructuring system. This phase margin approach provides a measure of how near to instability the ideal system is in terms of how much phase lag is permissible. This disadvantage of this approach is that the substructured system must be approximated to a linear transfer function. However, it can be readily used for the class of systems for which the DDE methods cannot or when it is impractical to use other techniques, giving it considerable advantages in terms of practical implementation. This technique is examined in § 3.3.

3.2 Delay differential equation models

The method we employ here is to model the substructured system with delay differential equations (DDEs), which are derived from the ODE model of the system by including explicitly the delay(s) due to the transfer system(s) [69]. A DDE model is a system of differential equations that depend on the current state of the system and on the state of the system some fixed time τ ago. As a general reference to the theory of DDEs see, for example, Diekmann et al. [92] or Stépan [93]. The advantage of DDE modelling is that we can use powerful analytical and numerical methods to

determine the stability of the DDE model and, hence, of the substructured system. Specifically, the loss of stability as a function of increasing delay is typically observed in substructured systems by the onset of oscillations. Because this corresponds in the DDE model to a pair of complex conjugate eigenvalues with zero real part, it is possible to determine the critical delay time τ_c, above which the system is unstable. Depending on the system under consideration, this can be done either by considering the characteristic equation for the eigenvalues of the DDE [92, 93] or with the numerical tool DDE-BIFTOOL [94].

To introduce the technique we use the single DOF example of § 2.3.1 and therefore restrict the system to the case of a single fixed delay τ. The origin of the delay in terms of the transfer system is identified in § 2.6 and we show in § 3.2.1 how the delay explicitly appears in the feedback force from the substructure. We can then use this DDE to ascertain stability in a number of different ways. Firstly, a perturbation analysis of the characteristic equation (under the assumption of small delay) can be used to compute an approximate expression of the critical time delay τ_c. Because the system in question is linear and quite simple, we can also compute explicit expressions for τ_c by computing the purely imaginary complex roots directly from the characteristic equation. In § 3.2.5 we extend this framework to consider several delays for the multi DOF example of § 2.3.2. Although it is still possible to compute the explicit solution for this linear system it becomes increasingly cumbersome. Therefore, we demonstrate how the stability regions can be computed numerically with the mathematical tool DDE-BIFTOOL [94] in order to validate the approach. Because this software does not require any special property of the governing equations, § 3.2.3 demonstrates how we can find τ_c in a general situation of more complex and nonlinear substructured systems.

Whatever the choice of the inner and outer-loop controllers the controlled system can again be modelled by a DDE, generally by modifying the DDE model which now contains additional parameters describing the controller. The stability of this new DDE model can be analysed efficiently for the dependence on the controller pa-

rameters. This knowledge can then be used to help develop robust control strategies which is studied in detail in § 5.5.

3.2.1 The substructured system

We consider the example of the single DOF mass-spring oscillator system of § 2.3.1. This well known linear system will allow us to demonstrate the fundamental problems associated with the occurrence of delay in a substructuring algorithm. From Figure 2.4 (§ 2.3.1), we see that the dynamics of the numerical model is governed by

$$m\ddot{z} + c(\dot{z} - \dot{r}) + k(z - r) = F, \tag{3.1}$$

where the feedback force, F, is the substructure response of $F = -k_s x$ and is treated as an external disturbance in the numerical model in order to simplify integration. As the transfer system has its own dynamics, it cannot react instantaneously and thus introduces the inevitable time delay. This means that

$$x(t) = az(t - \tau), \tag{3.2}$$

for some positive τ, where a is the amplitude accuracy of the closed loop response of the inner-loop controller. Note also that here we ignore any additional effects of physical disturbances and any transient nonlinear behaviour of the transfer system.

For the moment, assuming a unit gain response from the transfer system ($a = 1$), the delay τ introduces the systematic synchronisation error $z(t) - x(t) = z(t) - z(t - \tau)$ into the substructuring algorithm. Proposition 1 (§ 2.6, [75]) conjectures that, in general, the substructured system $z(t)$ approximates the emulated system $z^*(t)$ if this synchronisation error is small[1], that is, $z \to z^*$ if $x \to z$. Therefore, it is natural that the synchronisation error, e_2 (or *local* control error § 2.6), is a crucial measure for the accuracy of the substructuring experiment and, in fact, is the only explicit measurement of accuracy available for complex systems.

[1]This is the case in Hardware-in-the-loop testing, where because of the structure of the system being tested, the transfer systems have no dynamics.

We call the substructured system *unstable* if the synchronisation error grows exponentially in time and we call it *stable* if the synchronisation error remains bounded. As a result, when the synchronisation error is non-zero, the force is also described by a delayed state of the numerical model,

$$F = -k_s a z(t-\tau). \tag{3.3}$$

The overall substructured system is then governed by Eq. (3.1) with Eq. (3.3), which constitutes a delay differential equation (DDE) that can be written as

$$m\ddot{z} + c\dot{z} + kz + k_s a z(t-\tau) = c\dot{r} + kr\,. \tag{3.4}$$

We can now perform a detailed study of the substructured system Eq. (3.4) in order to determine the critical delay τ_c above which the system is unstable.

3.2.2 Explicit stability analysis

Using Eq. (3.4) with $r = \dot{r} = 0$ and $x(t) = az(t-\tau)$ we obtain the complimentary equation

$$m\ddot{z} + c\dot{z} + kz + k_s a z(t-\tau) = 0. \tag{3.5}$$

This can be expressed with non-dimensionalized parameters as

$$\frac{d^2 z}{d\hat{t}^2} + 2\zeta\frac{dz}{d\hat{t}} + z + pz(t-\tau) = 0, \tag{3.6}$$

where

$$\omega_n = \sqrt{\frac{k}{m}}, \qquad \hat{t} = \omega_n t, \qquad \hat{\tau} = \omega_n \tau, \qquad p = \frac{ak_s}{k}, \qquad \zeta = \frac{c}{2\sqrt{mk}}.$$

(Parameters values: $m_1 = 2.2$ kg, $k = k_s = 2250$ Nm^{-1} and $c = 15$ Nsm^{-1}.)

The introduction of a delay term into a linear ordinary differential equation (ODE) has two effects. First, it changes the spectrum of the ODE by a perturbation of order τ. Second, it introduces infinitely many new modes. If the delay is small, the new modes are all strongly damped and the perturbation of the ODE spectrum

can be expanded in the small parameter τ. This perturbation analysis is often easier to perform analytically than the search for all complex roots of the typically transcendental characteristic equation of the full DDE. For the parameters stated above the assumption that τ is small is valid. Thus, we pursue both approaches and compare the perturbation analysis with the full root analysis of the characteristic equation of Eq. (3.6).

Firstly, searching for solutions of the form $z = Ae^{\lambda\hat{t}}$, Eq. (3.6) can be written as

$$\lambda^2 e^{\lambda\hat{t}} + 2\zeta\lambda e^{\lambda\hat{t}} + e^{\lambda\hat{t}} + p(e^{\lambda\hat{t}}e^{-\lambda\hat{\tau}}) = 0, \tag{3.7}$$

which leads to the characteristic equation for the system of

$$\lambda^2 + 2\zeta\lambda + 1 + pe^{-\lambda\hat{\tau}} = 0. \tag{3.8}$$

The complex roots λ_i of Eq. (3.8) are the system eigenvalues, the sign of their real parts determines the stability of the system. The majority of large structures are lightly damped thus ζ is small. We may assume that $\hat{\tau}$ is small and expand $e^{-\lambda\hat{\tau}}$ to its first order extrapolation of $1 - \lambda\hat{\tau}$. Using this approximation, Eq. (3.8) becomes

$$\lambda^2 + \lambda(2\zeta - p\hat{\tau}) + (1 + p) = 0. \tag{3.9}$$

Solving for λ gives the roots

$$\lambda_{1,2} = -\frac{1}{2}(2\zeta - p\hat{\tau}) \pm \frac{1}{2}\sqrt{(2\zeta - p\hat{\tau})^2 - 4(1+p)}, \tag{3.10}$$

which governs the dominant eigenvalues for the DDE system given by Eq. (3.4) when $\tau \ll 1$. When $\hat{\tau} = 0$, Eq. (3.10) reduces to the eigenvalue equation for a standard underdamped spring-mass-damper system, for which the eigenvalues are complex and stable for positive values of m, c, p. Additionally, we note that ζ, ω_n and τ are positive quantities. Thus, because $\hat{\tau}$ is small, we can make the assumption that the eigenvalues remain complex, i.e. $4(1+p) > (2\zeta - p\hat{\tau})^2$. Therefore, the real parts of the eigenvalues from Eq. (3.10) determine the overall stability, such that the system is stable only if $p\hat{\tau} < 2\zeta$. Converted back to the original parameters this means that

the system is stable if the transfer system delay τ is less than the critical value

$$\tau_c = \frac{2\zeta}{p\omega_n} = \frac{c}{ak_s}. \tag{3.11}$$

This expression highlights that for lightly damped, stiff structures τ_c will be small and consequently the control algorithm must work harder to maintain stability. As the response delay, τ, increases past τ_c the real parts of the eigenvalues become positive and result in the transition of a pair of complex conjugate eigenvalues from the left to the right hand plane. This transition is called a Hopf bifurcation and entails the creation of (small) oscillations. Hopf bifurcations are a well know phenomenon in the context of DDEs [93, 95–97].

A previous analysis of the effect of time delay in substructuring [61] used an energy analysis of periodic orbits to equate the time delay to a form of negative damping. Eq. (3.11) clearly demonstrates how this negative damping manifests itself. In fact, the equivalent negative damping term can be expressed as $c_{neg} = -ak_s\tau$, with instability occurring at the point of sign change for the damping of the overall system. We note that as the transfer system response increasingly overshoots the numerical model demand the amount of negative damping also increases, and thus, reduces the margin to instability. Therefore, the amplitude accuracy of the transfer system response can also be seen as a form of negative damping, effectively stiffening the substructured system when it overshoots the demand.

The second approach to determining the stability boundaries of Eq. (3.6) is to search for points in the parameter space where the characteristic equation, Eq. (3.8), has purely imaginary solutions, that is, just undergoes a Hopf bifurcation. This analysis is valid not just for small τ but for any value of τ; see Gilsinn [96] and Larger and Goedgebuer [97] for similar approaches. Specifically, the stability boundaries are found in the parameter space by searching for solutions of the form $z = Ae^{j\omega t} = Ae^{j\hat{\omega}\hat{t}}$ where $\hat{\omega} = \frac{\omega}{\omega_n}$ is a positive real number (0 cannot be a characteristic root in this case). We insert $\hat{\omega}$ into the characteristic equation, Eq. (3.8), to obtain

$$-\hat{\omega}^2 + 2\zeta\hat{\omega}j + 1 + pe^{-j\hat{\omega}\hat{\tau}} = 0. \tag{3.12}$$

Splitting Eq. (3.12) into real and imaginary parts gives a system of two real equations:

$$0 = 1 - \hat{\omega}^2 + p\ \cos(\hat{\omega}\hat{\tau}) \qquad \text{for the real part,} \tag{3.13}$$

$$0 = 2\zeta\hat{\omega} - p\ \sin(\hat{\omega}\hat{\tau}) \qquad \text{for the imaginary part.} \tag{3.14}$$

We use the Equations Eq. (3.13) and Eq. (3.14) to express the parameters as functions of $\hat{\omega}$. In this way, we can identify all points in the parameter space where the DDE has purely imaginary eigenvalues and, thus, changes stability (at a Hopf bifurcation). Dividing Eq. (3.13) by Eq. (3.14) we get

$$\cot(\hat{\omega}\hat{\tau}) = \frac{\hat{\omega}^2 - 1}{2\zeta\hat{\omega}}. \tag{3.15}$$

The cot function is periodic, therefore

$$\hat{\tau} = \frac{1}{\hat{\omega}}\ \text{arccot}\left(\frac{\hat{\omega}^2 - 1}{2\zeta\hat{\omega}}\right) + \frac{2n\pi}{\hat{\omega}}, \tag{3.16}$$

where n is an integer, is satisfied on the stability boundary. If arccot is to be taken between 0 and π then n has to be non-negative since $\hat{\tau}$ is positive. Squaring and adding the equations Eq. (3.13) and Eq. (3.14) and rearranging for p (taking into account that p must be positive) we get

$$p = \sqrt{(\hat{\omega}^2 - 1)^2 + 4\zeta^2\hat{\omega}^2}. \tag{3.17}$$

Figure 3.1(a1) shows the curves for $n = 0$ to $n = 6$ (up to the limit of $\hat{\tau} = 30$) of the infinite solution set for the critical parameter pairs $(\hat{\tau}, p)$ in the $(\hat{\tau}, p)$-plane with ζ fixed at 0.1066. These curves are parameterized by $\hat{\omega}$ running from 0 to $+\infty$ in equations Eq. (3.16) and Eq. (3.17). Along these curves the system has a pair of purely imaginary eigenvalues and, hence, gains one additional unstable mode. Along the line $\hat{\tau} = 0$ the system is stable. Consequently, always the lowest parts of the curves define the stability boundary; the grey area is the region of stability. For comparison we have inserted into Figure 3.1(a1), as a dashed curve, the stability boundary obtained from the perturbation (approximate) analysis Eq. (3.11). As

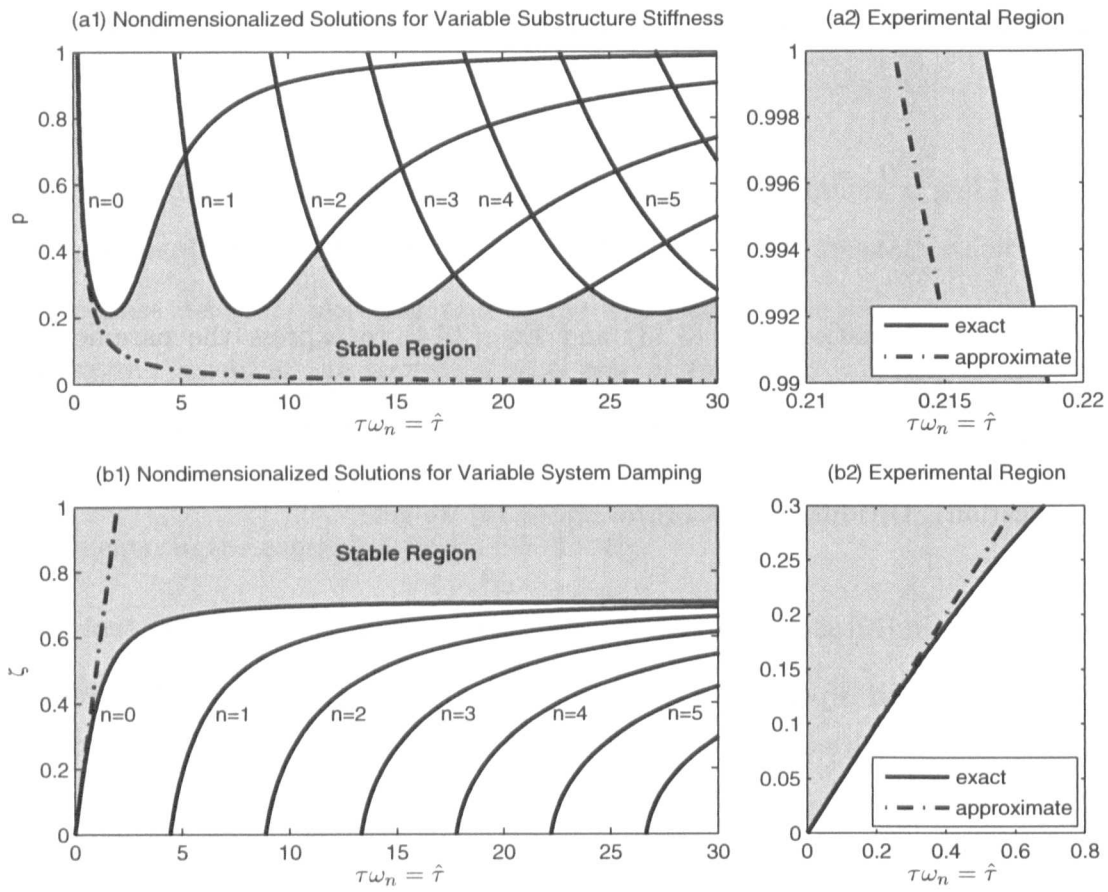


Figure 3.1: Non-dimensionalized Hopf stability boundaries of the DDE Eq. (3.6) for variable response delay τ. Fixed parameters: $\zeta = 0.1066$ in panels (a) and $p = 1$ in panels (b).

stated earlier, it can be seen that the approximation only holds for small values of the delay. Further more it is a slight underestimation of the explicit solution as the higher order terms are not included in the approximation. Figure 3.1(a2) shows an enlargement of the region where the ratio of the spring constants has a value of $p = 1$ for which the experimental testing is performed in § 3.4. The non-dimensionalized critical value $\hat{\tau}_c$ can be read off as $\hat{\tau}_c = 0.2165$, from which the critical time delay can be computed as $\tau_c = 6.77\,\text{ms}$.

To obtain the critical delay $\hat{\tau}_c$ and ζ for fixed p as parametric curves in the $(\hat{\tau}, \zeta)$-plane we rearrange Eq. (3.13) for $\hat{\tau}$ and Eq. (3.17) for ζ, thus expressing the critical

$\hat{\tau}$ and ζ as functions of $\hat{\omega}$ and p (taking into account that $\hat{\tau}$ and ζ must be positive):

$$\hat{\tau} = \frac{1}{\hat{\omega}} \arccos\left(\frac{\hat{\omega}^2 - 1}{p}\right) + \frac{2n\pi}{\hat{\omega}}, \tag{3.18}$$

$$\zeta = \frac{1}{2\hat{\omega}}\sqrt{p^2 - (\hat{\omega}^2 - 1)^2}, \tag{3.19}$$

where $\hat{\omega}$ runs from $\sqrt{\max\{0, 1-p\}}$ to $\sqrt{1+p}$, and n is any non-negative integer if arccos takes values between 0 and π. Figure 3.1(b1) shows the stability region (grey) and the critical values of $\hat{\tau}$ and ζ for fixed $p = 1$ using the curves defined parametrically by equations Eq. (3.18) and Eq. (3.19). The primary curve with $\hat{\tau}$ for $n = 0$ is always the stability boundary in the $(\hat{\tau}, \zeta)$-plane. Again, we have inserted into Figure 3.1(b1) as a dashed curve the approximate stability boundary given by Eq. (3.11) that was obtained from the perturbation analysis. This curve is only accurate for systems which are lightly damped with a maximum of approximately 15% damping for this structure, which can be seen from the enlargement of the experimental region in Figure 3.1(b2). Figures 3.1(a1) and 3.1(b1) also highlight that there are 'stable' parameter regions. These are regions where the system is stable regardless of the delay $\hat{\tau}$. We can compute the boundaries of these regions by rearranging Eq. (3.17) for $\hat{\omega}$,

$$\hat{\omega}^2 = 1 - 2\zeta^2 \pm \sqrt{(1 - 2\zeta^2)^2 + p^2 - 1}. \tag{3.20}$$

The right-hand-side of Eq. (3.20) is never real if $p < p_{\min} = 2\zeta\sqrt{1-\zeta^2} = 0.212$ (discriminant negative), giving rise to the stable region in Figure 3.1(a1). Moreover, the right-hand-side cannot be positive if $p \leq 1$ and $\zeta > \sqrt{2}/2 = 0.7071$, which accounts for the stable region in Figure 3.1(b1) for the specific case $k_s = k$ $(p = 1)$.

We note that the vast majority of structures, especially in the civil engineering field, are lightly damped such that operating in the region of stability for all $\hat{\tau}$ would be extremely unlikely. However, when substructuring mechanical components, such as in Chapter 6, other parts of the stable region are likely to be accessible.

3.2.3 Numerical stability analysis

For more complex DDEs than Eq. (3.6) it may become impossible to find stability regions as shown in Figure 3.1 by analytical calculations. We therefore move to a numerical approach for finding the stability regions when the substructured system is complex or nonlinear. In this section, we use a mathematical tool called DDE-BIFTOOL to demonstrate how this numerical analysis may be applied. DDE-BIFTOOL is a collection of Matlab routines for numerical bifurcation analysis of systems of DDEs with multiple fixed, discrete delays and is freely available for scientific purposes. A full description of the algorithms used within the DDE-BIFTOOL package and its limitations is given by Engelborghs et al. [94].

We used DDE-BIFTOOL to find the critical delay τ_c where the first Hopf bifurcations occurs and the substructured system destabilizes. Figure 3.2(a1) shows the real parts of the roots of the characteristic equation, Eq. (3.8), for the substructured system as the delay τ is increased (shown up to $\tau = 1$s). The system is stable when all roots are in the left half plane, that is, none of the curves are above zero. The first Hopf bifurcation takes place when the dominant branch crosses the dashed line where $\mathrm{Re}(\lambda) = 0$. As can be observed from the enlarged view in Figure 3.2(a2), stability is maintained until the response delay reaches the critical value of $\tau_c = 6.77$ms. This agrees with the value found in the explicit stability analysis in § 3.2.2.

The zero roots can be followed to see the effect of varying the structural parameters on the stability of the substructured system in terms of the critical delay. Figure 3.2(b) shows the stability boundary given by τ_c when the spring stiffness is varied. As expected, the boundary obtained with DDE-BIFTOOL agrees with that in Figure 3.1(a1). Note that the minimal value of the curves is observed at $k_s = 477$ such that $p = 0.212$. Figure 3.2(c) shows the stability region, bounded by τ_c, for changing system damping of the numerical model. Again, we have agreement with the result found from the characteristic equation shown in Figure 3.1(b1). As the damping coefficient increases to $c = 99.5$ we see that the stability boundary reaches

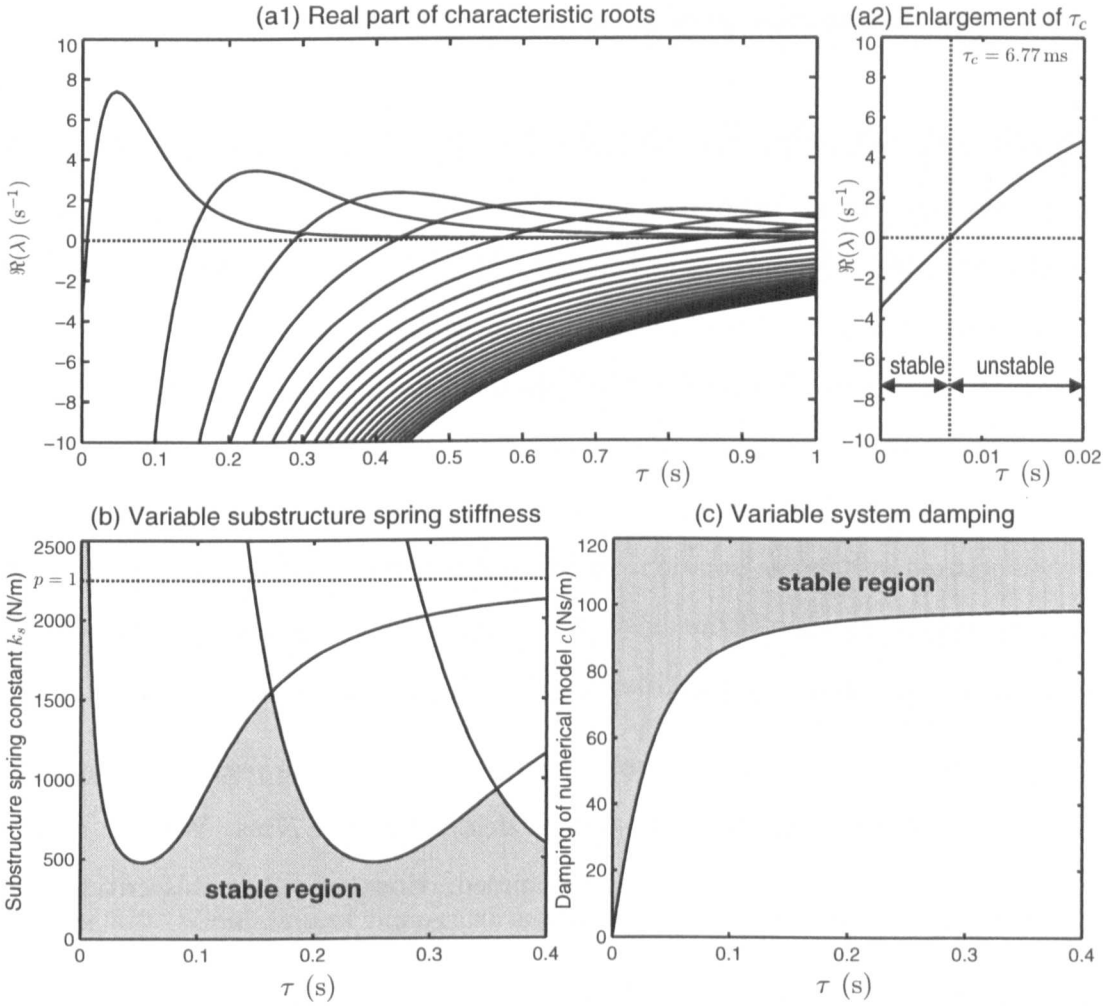


Figure 3.2: (a1) Real part of complex roots of the substructured system computed using DDE-BIFTOOL with enlargement of the critical region (a2). Hopf bifurcation diagram showing the stability region for, (b) variable substructure spring stiffness k_s ($c = 15$Ns/m), and (c) variable system damping c ($k_s = 2250$N/m). Other system parameters: $m = 2.2$kg and $k = 2250$N/m.

an asymptote corresponding to a system damping of $\zeta = 0.7071$.

Overall, the results in this section agree with the explicit stability analysis in § 3.2.2. This demonstrates the potential of the numerical stability analysis with DDE-BIFTOOL, with the added advantage that it works also for far more complex and nonlinear systems. This will be further examined in § 3.2.5 and § 5.5.

3.2.4 Simulation of a pure delay

To demonstrate how the delay induced instability analysed in § 3.2.2 and § 3.2.3 manifests itself in an experiment, we first produce time domain and frequency response diagrams in numerical simulation. These simulations will allow us to idealize the dynamics of the transfer system as a pure unit gain response with a constant time delay. This approach isolates the delay induced error which causes the instability from other parasitic experimental effects, such as static friction, backlash and noise. This allows us to observe some of the characteristic features of the substructured system in isolation. It also gives us full control over the structural parameters and the size of the delay. In the simulations Eq. (3.4) is integrated by a fourth-order Runge-Kutta algorithm with a step size of 1ms.

Figure 3.3(a) shows a simulated substructure test where the response delay is $\tau =$ 7ms, which is just larger than the critical delay of $\tau_c = 6.77$ms. When $\tau < \tau_c$ the numerical model error is "small" and bounded. However, when this critical value of $\tau_c = 6.77$ms is exceeded then the error is unbounded and grows exponentially — the overall system damping is effectively negative [61]. This situation is shown in Figure 3.3(b), ignoring the additional transient effects at the start of the simulation. Since the response delay is only just larger than τ_c in this instance, the growth coefficient is very small. We can observe the growth coefficient from Figure 3.2(a). In this case the magnitude of the eigenvalue at the specific delay is $\Re(\lambda) = 0.105$ at $\tau = 0.007s$.

Additionally, we observe that the frequency at which instability occurs is constant and independent of the excitation frequency. Figure 3.4 shows the magnitude of the numerical model response over the experimental range of frequencies for an increasing simulated response delay τ for a test duration of 5 seconds. The excitation frequency in this example is 3Hz at constant amplitude. Instability occurs at a frequency of approximately 7.1Hz and grows with the expected exponential coefficients for varying τ. We can approximate the instability frequency, ω_I, from

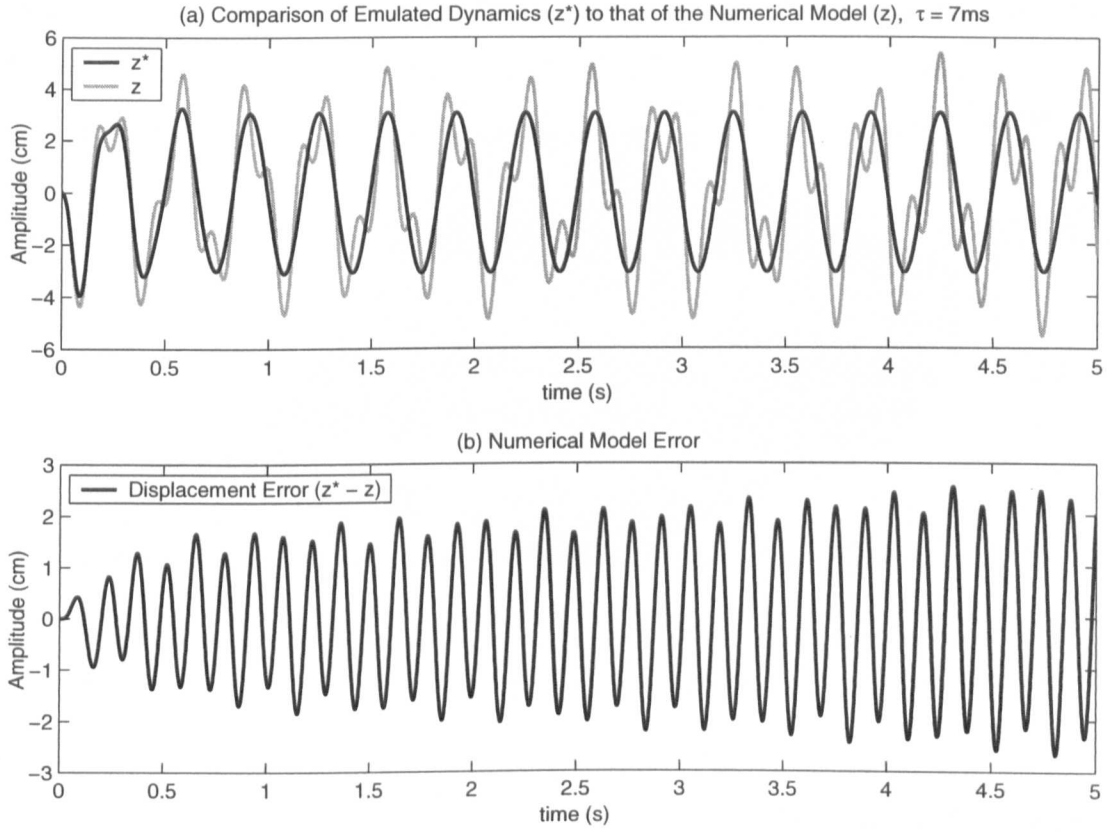


Figure 3.3: Simulation of numerical model accuracy caused by a transfer system delay of $\tau = 7$ms in the substructuring algorithm (where, $\tau_c = 6.77$ms).

Eq. (3.10) using the perturbation analysis at the point of instability, $p\hat{\tau} = 2\zeta$, such that $\lambda_i = j\sqrt{1+p}$. Removing the non-dimensionalization,

$$\omega_I = \lambda_i \omega_n = \sqrt{\frac{k+k_s}{m}} = 7.198Hz, \tag{3.21}$$

for this experimental setup. We can find a more accurate value for this instability frequency from the complex root solutions at $n = 0$. From Equation (3.20) for the case where $p = 1$ and at the point of instability,

$$\omega_I = \omega_n \sqrt{(2-4\zeta^2)^2} = 7.116Hz. \tag{3.22}$$

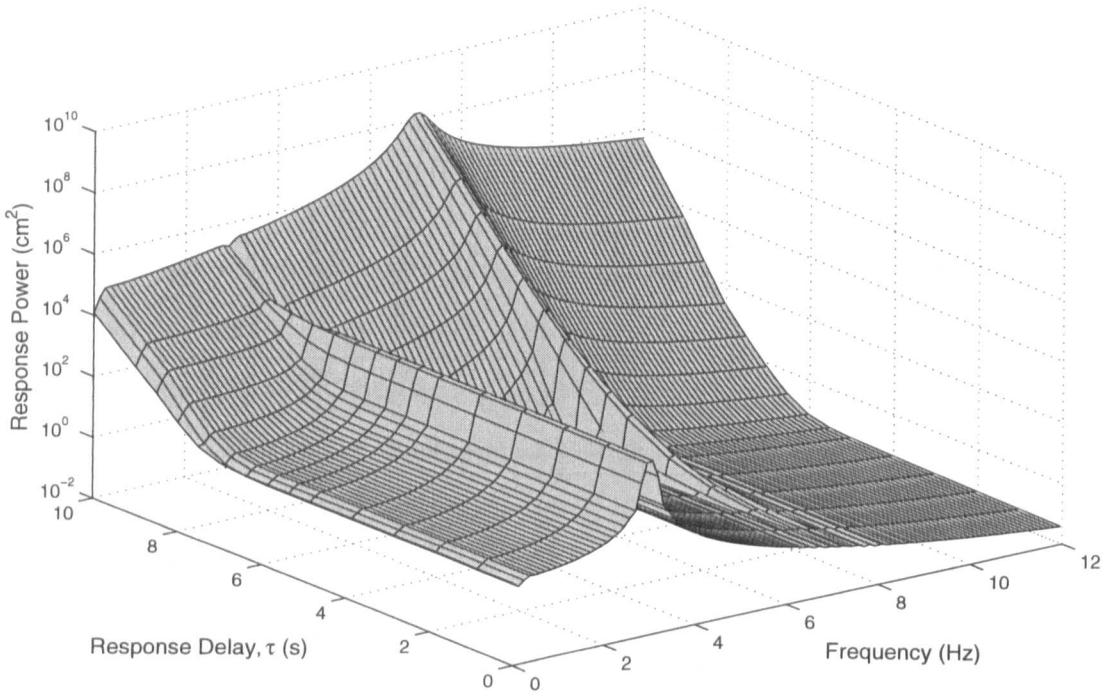


Figure 3.4: Exponential growth of instability independent of the 3Hz excitation frequency (5s test data).

3.2.5 Multi DOF analysis

We now consider the example of the multi DOF mass-spring oscillator system of § 2.3.2. The substructured system is shown in Figure 2.6 and the dynamics of the numerical model given by Eq. (2.6). Due to the fact we can write down explicit equations of motion for the substructure in this case, we know the emulated force feedback, $F^* = [F_1^* \quad F_2^*]^T$, at every time interval.

For this analysis, we make the assumption that the two transfer systems have identical dynamics such that there is only one fixed delay, τ, in the system and that we achieve a unit gain response from each transfer system. Therefore, $x_i = z_i(t-\tau)$ where $i = 1, 2, 3$ which gives an actual force feedback of

$$\begin{aligned} F_1 &= c_{31}[\dot{z}_3(t-\tau) - \dot{z}_1(t-\tau)] + k_{31}[z_3(t-\tau) - z_1(t-\tau)], \\ F_2 &= c_{32}[\dot{z}_2(t-\tau) - \dot{z}_3(t-\tau)] + k_{32}[z_2(t-\tau) - z_3(t-\tau)], \end{aligned} \tag{3.23}$$

from Eq. (2.7). The overall substructured system is then governed by Eq. (2.6) with Eq. (3.23), which constitutes a coupled delay differential equation (DDE) that can be written as

$$\begin{aligned} m_1\ddot{z}_1 + c_1\dot{z}_1 + k_1 z_1 + c_{31}[\dot{z}_1(t-\tau) - \dot{z}_3(t-\tau)] + k_{31}[z_1(t-\tau) - z_3(t-\tau)] \\ = c_1\dot{r}_1 + k_1 r_1, \\ m_2\ddot{z}_2 - c_2\dot{z}_2 - k_2 z_2 - c_{32}[\dot{z}_2(t-\tau) - \dot{z}_3(t-\tau)] + k_{32}[z_2(t-\tau) - z_3(t-\tau)] \\ = -c_2\dot{r}_2 - k_2 r_2. \end{aligned} \tag{3.24}$$

We can now study the substructured system of Eq. (3.24) in order to determine the critical delay τ_c above which the system is unstable. However, due to the damper components in the substructure, which will also be affected by the transfer system delay, this becomes a second-order DDE. Additionally, without making the assumption that the middle mass is stationary, such that $z_3 = \dot{z}_3 = 0$, the explicit analysis becomes complex and impractical, [98].

We therefore remove the need to make any simplifications by again utilizing the mathematical tool DDE-BIFTOOL ([94]) to find the critical delay τ_c. For simplicity we study the symmetrical system of $m = m_1 = m_2 = m_3 = 2.2$, $k = k_1 = k_2 = k_{31} = k_{32} = 4750$N/m and $c = c_1 = c_2 = c_{31} = c_{32} = 6$Ns/m. The actual values for the system parameters makes no difference in the application of DDE-BIFTOOL, just the number of delay within the system. Figure 3.5 shows the real parts of the roots of the characteristic equation, Eq. (3.24), for the substructured system as the delay τ is increased (shown up to $\tau = 0.05$s). As before, the system is stable when all roots are in the left half plane (none of the curves are above zero), therefore instability occurs at the first Hopf bifurcation — when the dominant branch crosses the dashed line where $\mathrm{Re}(\lambda) = 0$. As can be observed from the enlarged view in Figure 3.5(b), stability is maintained until the response delay reaches the critical value of $\tau_c = 2.502$ms.

We note that this stability criterion is only valid for case where the *actual* transfer system delays, τ_1 and τ_2, are equal. The analysis, by DDE-BIFTOOL, could now be

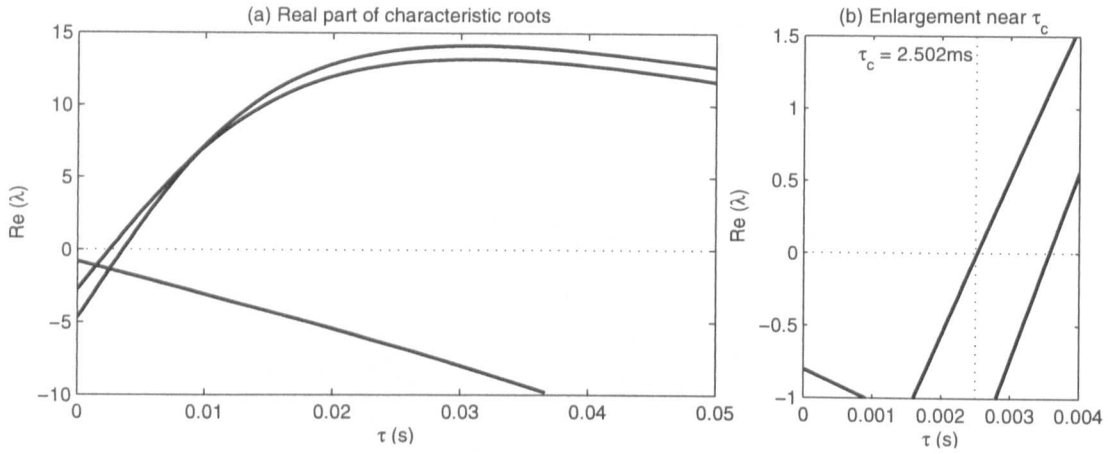


Figure 3.5: (a) Real part of complex roots of the substructured system computed using DDE-BIFTOOL with enlargement of the critical region (b).

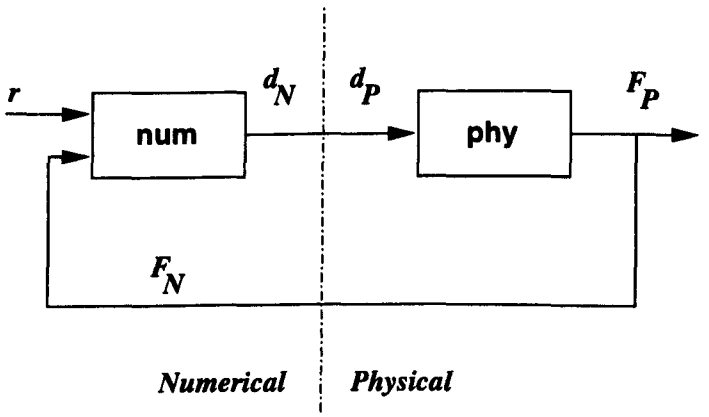
extended to encompass the substructured system with independent variable delays such that $\tau_1 \neq \tau_2$; this is a potential line of future research, see § 7.2.

3.3 Phase margin approach

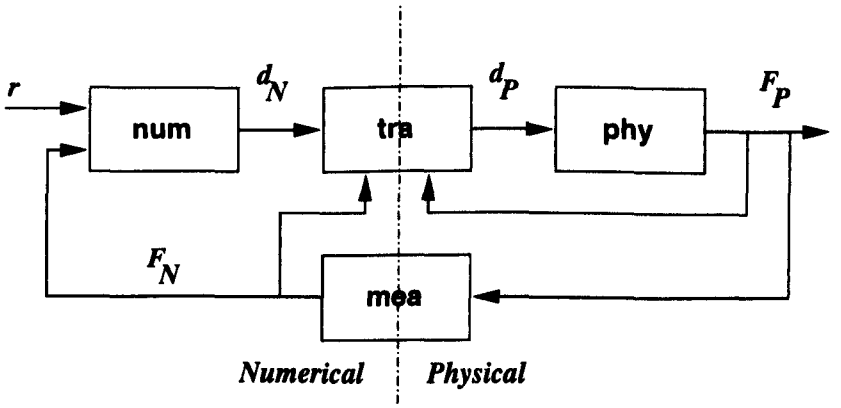
As pointed out previously in Chapter 2, substructuring can be viewed as a control problem. However, unlike conventional control system design which aims for a well-damped closed-loop system, the corresponding substructuring design often has lightly-damped behaviour near the boundary of stability. However, because the substructuring testing technique has been developed primarily from a civil engineering perspective [61, 69, 73], stability has not been studied using a control theoretic approach. In this section we apply tools from control theory to study the relative and robust stability again for the single DOF substructured system of § 2.3.1. In particular we focus on the effect of unmodelled delays and other uncertainties which occur during substructuring. This technique was originally presented by Gawthrop et al. [74]. We present an alternate technique for calculating the critical delay, τ_c, to that of using DDE models which can be used, although only a linear approximation, when it is not possible or impractical to apply the DDE techniques.

3.3.1 The substructured system

Following on from Figure 2.4 (§ 2.3.1) of the single DOF substructuring example, Figure 3.6 shows two block diagram representations of substructuring. Figure 3.6(a) shows the ideal case where a numerical model, **num**, is coupled directly to a substructure, **phy**, and there are no transfer system dynamics and hence no need for a controller. The detail of the two substructure blocks is: for **phy** the output F_P and input d_P are a collocated force and displacement pair connecting **phy** to the numerical part of the substructured system. Similarly for **num** the input F_N and output d_N are the collocated force and displacement pair connecting **num** to the physical part of the substructured system. The signal r represents the net effect of



(a) Ideal substructured model



(b) Practical substructured model

Figure 3.6: Substructuring: block diagram approach

external forces in the numerical part of the model.

In this ideal situation, $F_N = F_P$ and $d_P = d_N$ and the dynamic behaviour of the substructured system is exactly replicated that of the emulated system, as stated in Proposition 1. However, for structural or mechanical systems, the physical system input d_P has to be generated by a transfer system which has the control objective of setting $d_P \approx d_N$. The physical force F_P is measured by a *sensor system* which also has it's own dynamics. In practice the ideal sensor system has the relationship that $F_P \rightarrow F_N$ as $d_p \rightarrow d_N$.

Figure 3.6(b) gives a block diagram representation of the practical case. In addition to the two blocks of Figure 3.6(a); **tra** represents the controlled transfer system including both inner- and outer-loop control systems, and **mea** represents the measurement sensor system, which includes the force transducer and associated power supplies (this is assumed not to interact with **phy**). We note that physical part of **tra**, $T_P(s)$, usually consists of the actuator and inner-loop controller and is affected by F_P (an actuator only has a finite performance capacity envelope in which it will operate in a linear fashion), while the numerical (augmented) part of **tra**, $T(s)$, is the outer-loop controller with its accuracy being affected by the measured version of F_P (F_N). At this point, the following assumption is made

Assumption 1 *The four systems in Figure 3.6(b) are* linear, time-invariant *and* stable.

Using Assumption 1, the ideal substructuring case, Figure 3.6(a), may be represented by:

$$F_P = P(s)d_P \qquad \textbf{(phy)}, \tag{3.25}$$

$$d_N = N_1(s)r - N_2(s)F_N \tag{3.26}$$

$$= N(s)(N_r(s)r - F_N) \qquad \textbf{(num)}, \tag{3.27}$$

where, $P(s)$ is the transfer function corresponding to **phy**, $N_1(s)$ and $N_2(s)$ in Eq. (3.26) are separate parts of the numerical model, which we re-express in Eq. (3.27)

in a more convenient form for later analysis; $N(s)$ is the transfer function corresponding to **num** and $N_r(s)$ is a transfer function representing the interface between **num** and the external forcing. In the ideal case there are no transfer system or measurement dynamics, such that $F_N = F_P$ and $d_P = d_N$. Then equations Eq. (3.25) and Eq. (3.27) for the physical substructure and the numerical model dynamics may be simplified such that the overall system dynamics are identical to that of the emulated system. This leads to the relation

$$F_P = \frac{L_0(s)}{1 + L_0(s)} N_r(s) r, \tag{3.28}$$

where $L_0(s) = P(s)N(s)$ and is defined as the *nominal* loop gain.

For the realistic substructuring representation, Figure 3.6(b), the dynamics of the transfer system and the measurement system must be included. We define these dynamics as

$$d_P = T(s)d_N - T_P(s)F_P \qquad \textbf{(tra)}, \tag{3.29}$$

$$F_N = M(s)F_P \qquad \textbf{(mea)}, \tag{3.30}$$

where the term $T_P(s)F_P$ of Eq. (3.29) includes the *net* effect of F_P and F_N on **tra**, and $M(s)$ is the transfer function representing the measurement sensor system. From Assumption 1, each transfer function explicitly appearing in equations Eq. (3.25)–Eq. (3.30) is *stable*. The issue is then to investigate whether the dynamics of the substructured system shown in Figure 3.6(b) is also stable.

Rearranging equations Eq. (3.25), Eq. (3.27), Eq. (3.29) and Eq. (3.30) the representation of Figure 3.6(b) may be written as

$$[1 + P(s)T_p(s)]F_p = L_0(s)[T(s)N_r(s)r - T(s)M(s)F_p]. \tag{3.31}$$

Defining the *neglected* gain as $\Lambda(s)$ and the *neglected* forward gain as $\Lambda_r(s)$ we obtain

$$\Lambda(s) = [1 + P(s)T_P(s)]^{-1} T(s)M(s), \tag{3.32}$$

$$\Lambda_r(s) = [1 + P(s)T_P(s)]^{-1} T(s)). \tag{3.33}$$

Defining an *equivalent* force, $F_e = \Lambda_r(s)N_r(s)r$, the system dynamics may be expressed as

$$F_P = L_0(s)[F_e - \Lambda(s)F_P]. \tag{3.34}$$

Figure 3.6(b) can thus be represented by the classical feedback system of Figure 3.7.

Finally, defining $D(s)$ as the transfer function relating F_e and F_P for the practical substructured system, using Eq. (3.34) we can write:

$$D(s) = \frac{L_0(s)}{1 + \Lambda(s)L_0(s)}, \tag{3.35}$$

such that

$$F_P = D(s)F_e. \tag{3.36}$$

Using Eq. (3.28) and recognising that for the ideal case $\Lambda(s) = \Lambda_r(s) = 1$, the corresponding nominal transfer function relating to the ideal substructured system, and the emulated system, may be defined as

$$D_0(s) = \frac{L_0(s)}{1 + L_0(s)}. \tag{3.37}$$

In Chapter 5 we discuss the use of different robustness compensation schemes. We note that $D_0(s)$ will explicitly include these algorithms and therefore change. Thus for comparison, we define $D_{em}(s)$ as the emulated system transfer function such that $D_{em}(s) = D_0(s)$ when $L_0(s)$ does not incorporate any compensation schemes.

3.3.2 Relative and robust stability

Figure 3.7 and the corresponding closed-loop system Eq. (3.36) are in the classical feedback control system form where $L_0(s)$ would be interpreted as the "system" and $\Lambda(s)$ as the "controller". This means that a range of standard control system design techniques ([99]) can be brought to bear on the problem.

For example, *relative stability* ([99]) can be characterized as follows. Define the *critical frequency* ω_c as the solution of

$$|L(j\omega_c)| = 1, \tag{3.38}$$

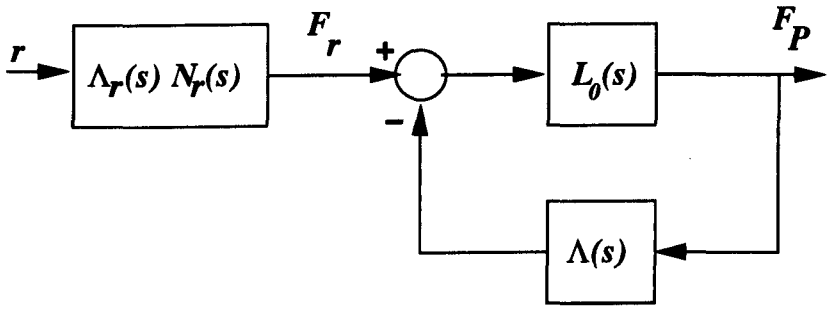


Figure 3.7: Sensitivity feedback system

where the *actual* loop gain, $L(s)$, is defined as

$$L(s) = \Lambda(s)L_0(s). \tag{3.39}$$

The corresponding *phase margin* ϕ_m may be written as

$$\phi_m = \pi - \angle L(j\omega_c). \tag{3.40}$$

The phase margin provides a measure of how near the ideal system (with $\Lambda(s) = 1$) is to instability in terms of how much phase lag (due to $\Lambda(s) \neq 1$) is permissible. For example, if the neglected dynamics comprise a pure delay ($\Lambda(s) = e^{-s\tau}$) then the *critical delay*, τ_c, is the time delay which would give a phase lag of ϕ_m and is given by

$$\tau_c = \frac{\phi_m}{\omega_c}. \tag{3.41}$$

This gives an alternative method for computing the critical delay, in addition to that developed by Wallace et al. [69] which uses DDE models presented in § 3.2. For the class of system for which the DDE methods cannot be applied or when it is impractical to use the technique, Eq. (3.41) could still be used in many cases (even if only as a linear approximation) to give an estimate of τ_c. The link between the two techniques is discussed further in § 3.3.3.

However, for substructuring systems we would like to apply more general, *robust stability* methods. Using the approach outlined in Goodwin et al. [99, sec. 5.9], together with the assumption that both $L_0(s)$ and $\Lambda(s)$ are stable implies that the

closed-loop system of Figure 3.7 is stable if

$$|D_0(j\omega)|\,|\Delta(j\omega)| \leq 1 \quad \forall\omega, \tag{3.42}$$

where

$$\Delta(s) = \Lambda(s) - 1. \tag{3.43}$$

This is a conservative result but has the advantage of bounding the error transfer function Δ in terms of the desired system $D_0(j\omega)$. In particular is shows that $\Delta(j\omega)$ must be small at those frequencies where $D_0(j\omega)$ is large — typically at the resonant frequencies of the desired system. Although these methods are standard in the control system context, they are novel in the substructuring context.

3.3.3 Explicit stability analysis

The single DOF example of a substructured system is shown schematically in § 2.3.1. The system has a numerical substructure consisting of a mass of $m$kg, a spring with stiffness kN/m and damper with constant cNs/m and a physical substructure consisting of a spring with stiffness k_sN/m. In this case

$$L_0(s) = \frac{k_s}{ms^2 + cs + k}, \tag{3.44}$$

$$N_r(s) = cs + k. \tag{3.45}$$

Defining the natural frequency of the numerical subsystem as $\omega_n = \sqrt{\frac{k}{m}}$, the corresponding damping ratio $\zeta = \frac{c}{2m\omega_n}$ and $p = \frac{k_s}{k}$ we can write

$$L_0(s) = \frac{p\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}, \tag{3.46}$$

$$N_r(s) = m(2\zeta\omega_n s + \omega_n^2). \tag{3.47}$$

Defining $\hat{s} = \frac{s}{\omega_n}$, this can be rewritten in normalised form as

$$L_0(s) = \frac{p}{\hat{s}^2 + 2\zeta\hat{s} + 1}, \tag{3.48}$$

$$N_r(s) = k(2\zeta\hat{s} + 1). \tag{3.49}$$

Defining $\hat{\omega} = \frac{\omega}{\omega_n}$ and using Eq. (3.38), the critical frequency corresponding to Eq. (3.48) is the solution of

$$(1-\hat{\omega}^2)^2 + 2\zeta^2\hat{\omega}^2 - p^2 = \hat{\omega}^4 + (4\zeta^2 - 2)\hat{\omega}^2 + (1-p^2) = 0. \tag{3.50}$$

Eq. (3.50) is quadratic in $\hat{\omega}^2$. It has real solutions if

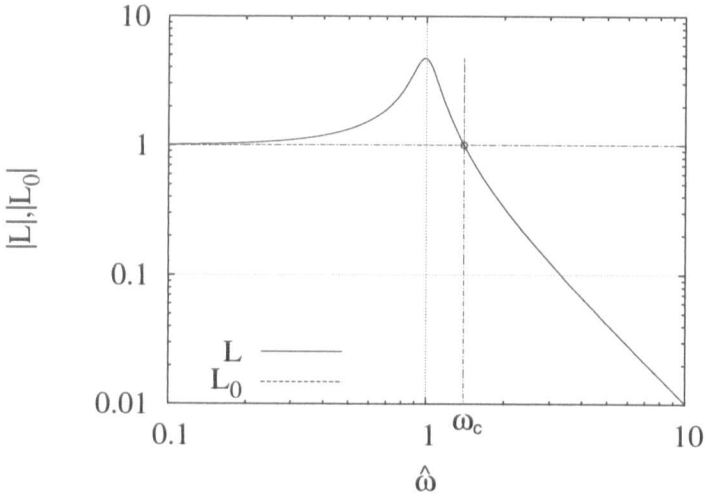
$$(4\zeta^2 - 2)^2 > (1-p^2). \tag{3.51}$$

There are two cases: if $p \geq 1$ then Eq. (3.51) is independent of ζ, otherwise the condition depends on the value of ζ. In the case of real solutions, the positive square root of a positive solution gives a (positive) value of $\hat{\omega}$ satisfying Eq. (3.38).

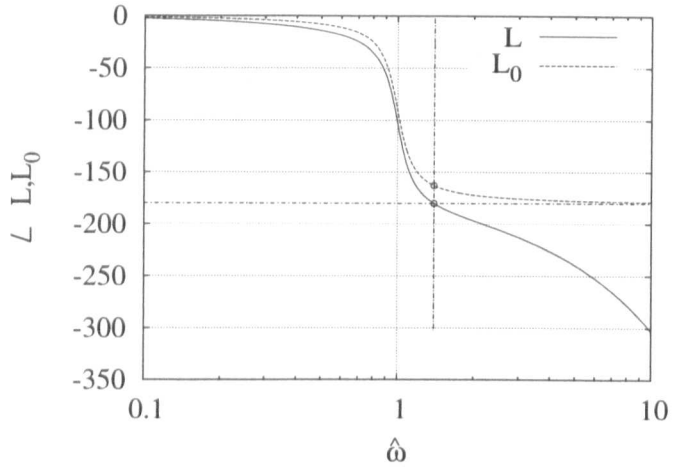
Figure 3.8(a) shows $\log|L_0(j\hat{\omega})|$ plotted against $j\hat{\omega}$ for $p = 1$ and $\zeta = 0.1066$. In this case, $\hat{\omega}_c = 1.398$, the frequency at which $|L_0(j\hat{\omega})| = 1$. Figure 3.8(b) gives the corresponding phase (in degrees) indicating a phase margin of $17.34° = 0.3027$rad. This gives a critical delay of $\hat{\tau}_c = 0.2165$ ($\tau_c = 6.77$ms), which is precisely the value obtained from the DDE numerical analysis of § 3.2.2, confirming the fact that both methods are exact.

The corresponding diagrams for $L = e^{-j\hat{\omega}}L_o$ are plotted in Figures 3.8(a) and 3.8(b). As predicted, $L(j\hat{\omega}_c) = -1$ corresponding to $|L(j\hat{\omega}_c)| = 1$ and $\angle L(j\hat{\omega}_c) = -180°$. Figure 3.9(a) shows how the phase margin varies with p and Figure 3.9(b) how it varies with ζ. For small values of ζ, the phase margin ϕ_m is approximately proportional to ζ. Larger values of p give a reduced phase margin. Figure 3.8(a) also provides alternative insight into the solution of Eq. (3.50). From Eq. (3.48), the sole effect of p is to move the curve of Figure 3.8(a) vertically. It is therefore clear that when $p > 1$, $\log|L_0(j\hat{\omega})| = 0$ at only one frequency implying a single positive real solution of Eq. (3.50) for $\hat{\omega}^2$. On the other hand, if $p < 1$ there is no solution if the peak of the curve is below zero. From Eq. (3.48), the peak value occurs at an approximate frequency of $\hat{\omega} = 1$ where $|L_0(j\hat{\omega})| = \frac{p}{2\zeta}$. Thus, as also indicated in Figure 3.9(a), the phase margin is infinite when $p < 2\zeta$.

The robustness criterion Eq. (3.42) can be examined by plotting both $\frac{1}{|D(j\hat{\omega})|}$ and

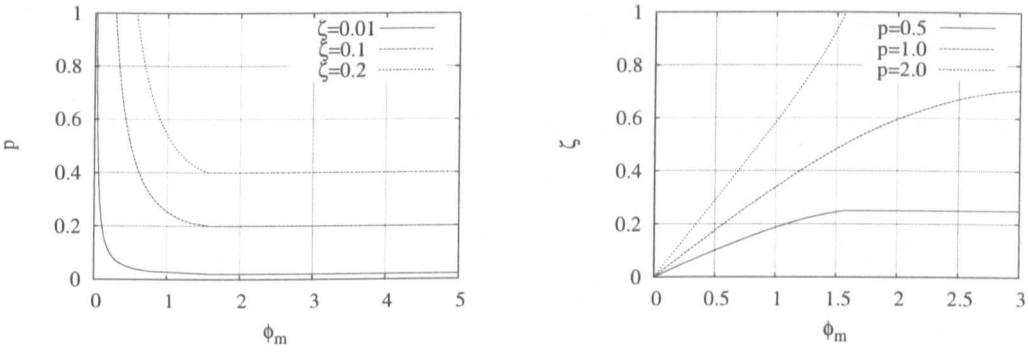


(a) $\log|L_0(j\hat{\omega})|$ & $\log|L(j\hat{\omega})|$ v. $\log\hat{\omega}$



(b) $ang.L_0(j\hat{\omega})$ & $ang.L(j\hat{\omega})$ v. $\log\hat{\omega}$

Figure 3.8: Sensitivity: Phase margin. (a) shows the log magnitude of the nominal $\log|L_0|$ and actual $\log|L|$ plotted against log normalised frequency $\log\hat{\omega}$; because there is only a phase error (pure delay), the curves are the same; the critical frequency $\hat{\omega}_c$ is marked by a vertical line and a unit gain by a horizontal line. (b) shows the corresponding phases together with a vertical line at the critical frequency $\hat{\omega}_c$ and a horizontal line at $-180°$. The phase margin is the vertical distance between $-180°$ and the corresponding phase curve $ang.L_0(j\hat{\omega})$ at $\hat{\omega} = \hat{\omega}_c$. In this case, the actual system is such that the neglected time delay is on the boundary of stability.



(a) p v. $\phi_m(rad)$ (b) ζ v. $\phi_m(rad)$

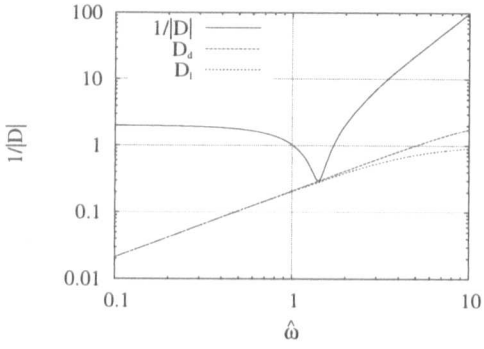
Figure 3.9: Sensitivity: dependence on parameters; (a) shows how the phase margin ϕ_m depends on p for three values of ζ (in this case, the phase margin ϕ_m *decreases* with p); (b) shows how the phase margin ϕ_m depends on ζ for three values of p (in this case, the phase margin ϕ_m *increases* with ζ).

$|\Delta(j\omega)|$ on the same diagram. For example, Figure 3.10(a) shows $\frac{1}{|D(j\hat{\omega})|}$ when $p = 1$ and $\zeta = 0.1066$. On the same diagram, $|\Delta(j\omega)|$ is plotted for two cases:

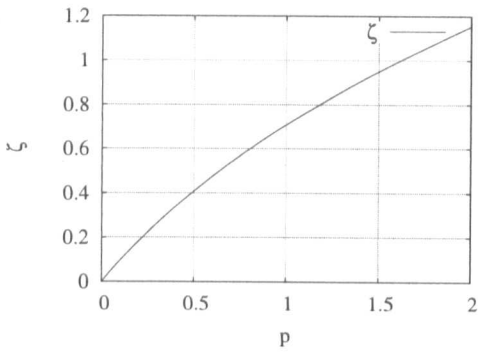
$$\Lambda(\hat{s}) = \begin{cases} e^{-\hat{\tau}\hat{s}}, & \text{(shown as } \Delta_d \text{ in Figure 3.10(a))} \\ \frac{1}{1+\hat{\tau}\hat{s}}, & \text{(shown as } \Delta_l \text{ in Figure 3.10(a))} \end{cases} \tag{3.52}$$

where $\hat{\tau} = 0.21$. In this case, stability is predicted in each case as Eq. (3.42) is satisfied. However, this would not be the case if τ were increased slightly. Note that both forms of $\Lambda(\hat{s})$ of Eq. (3.52) give similar results in this case indicating that *phase* error is more important that *amplitude* error in this case. Note that $\hat{\tau}_c = 0.21$ predicted by this (conservative) robust stability method is less than that predicted by the the exact relative stability (phase margin) approach of $\hat{\tau}_c = 0.2165$. However, the robustness approach is more general in that the uncertainty does not need to be parameterized by a transfer function.

The minimum value of $\frac{1}{|D(j\hat{\omega})|}$ occurs at $\hat{\omega}^2 \approx (1+p)$ with a value of approximately $\frac{2\zeta\sqrt{1+p}}{p}$. Noting that the maximum value of $\Delta = 1 - e^{-\hat{\tau}\hat{s}}$ is 2, it follows that the



(a) Robust stability



(b) Stability for any delay

Figure 3.10: (a) Shows the inverse magnitude of $D(j\hat{\omega})$ against $j\hat{\omega}$ on a logarithmic scale. For comparison two possible uncertainty transfer functions $e^{-j\hat{\omega}\hat{\tau}}$ and $\frac{1}{1+j\hat{\omega}\hat{\tau}}$ are plotted for $\hat{\tau} = 0.21$. (b) Shows the asymptotic stability of the substructured system.

substructured system will be stable for *any* delay τ if

$$\zeta > \frac{p}{\sqrt{1+p}}, \tag{3.53}$$

thus agreeing with the DDE analysis. Figure 3.10(b) shows the boundary implied by Eq. (3.53).

3.4 An experimental substructuring example

The experimental transfer system equipment has been extensively analysed and is described in § 2.3.4. It is known that a good model for the nominal dynamics of the system under an inner-loop proportional (P) control with $k_p = 1$ can be approximated to a delay of $\tau \approx 9.4$ms for this excitation frequency. Both § 3.2 and § 3.3 confirm that the critical value of stability for the single DOF system analysed is $\hat{\tau}_c = 0.2165$ or $\tau_c = 6.77$ms. As $\tau > \tau_c$, some form of delay compensation must be applied to the substructuring algorithm to achieve a stable algorithm. The delay compensation techniques used in this work are discussed in full in Chapter 4 but are required here in order to demonstrate the stability characteristics of an experimental

substructured system. To this end, we use a forward prediction algorithm (§ 4.3.5) which allows us to predict the numerical model forward a given magnitude of time, and thus, reduce the transfer system delay by the same amount. Predicting forward 9.4ms would result in the transfer system delay $\tau \approx 0$.

Figure 3.11 shows the experimental results for a wall excitation of 3Hz and constant delay compensation (§ 4.3.5) of 9.4ms. It can be seen from Figure 3.11(a) that the numerical model dynamics z closely replicate those of the emulated system z^*, losing accuracy briefly at direction change for the actuator. Note that the transfer system dynamics are not shown on this plot but are represented by the synchronisation subspace plot Figure 3.11(b) (see § 2.5 [91]). Perfect synchronisation is represented by a straight diagonal line. A constant delay turns this straight line into an ellipse, as can be seen from the limit of stability shown in grey representing z vs. $z(t-\tau_c)$. We can see from the subspace plot that there is generally a high level of synchronisation, well below the stable limit, apart from when the actuators change direction. Here we observe a region of loss in accuracy as the control signal must reach a certain level

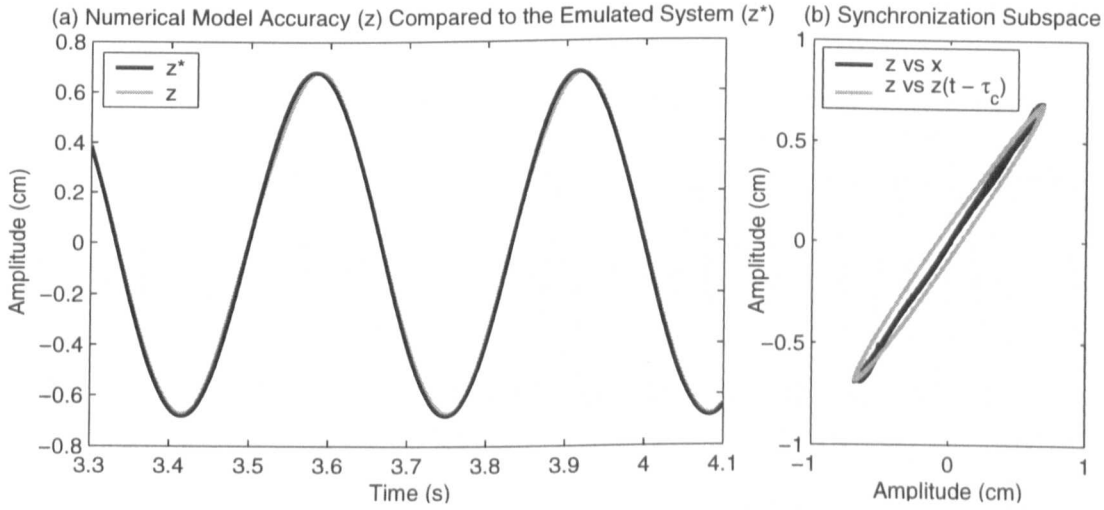


Figure 3.11: Experimental real-time dynamic substructure test with wall excitation of 3Hz and delay compensation of 9.4ms. Transfer system synchronisation is shown in (b) with limit of stability represented by the z vs $z(t-\tau_c)$ loop. τ_c=6.77ms.

to overcome the static friction of the actuator internal mechanisms before the piston will move (the *deadzones* as described in § 2.3.4). In fact, the algorithm verges into the unstable region at both limits. However, despite this, once the static friction is overcome, synchronisation is quickly regained. This shows that the instability shown by a substructured system may not be not catastrophic. For instability to grow cumulatively, the synchronisation must remain in the unstable region for a longer period of time, that is a number of successive time steps. Therefore, if the control algorithm can recover more quickly than the exponential growth (given by the real part of the eigenvalue at the actual delay as can be observed from Figure 3.2), then the system will be able to recover regardless of the disturbance.

The transition to instability can be seen from Figure 3.12 for this experimental system. It is qualitatively similar to the simulated result shown in Figure 3.3 and occurs at a forward prediction magnitude of 2.6ms. The actual response delay of the transfer system is approximately $\tau = 9.4$ms for this excitation condition, giving an experimental limiting value of $\tau_c \approx 6.8$ms. The frequency at which instability is observed is shown in Figure 3.13; it again corroborates the results of § 3.2.4 that $\omega_I \approx 7.1$Hz.

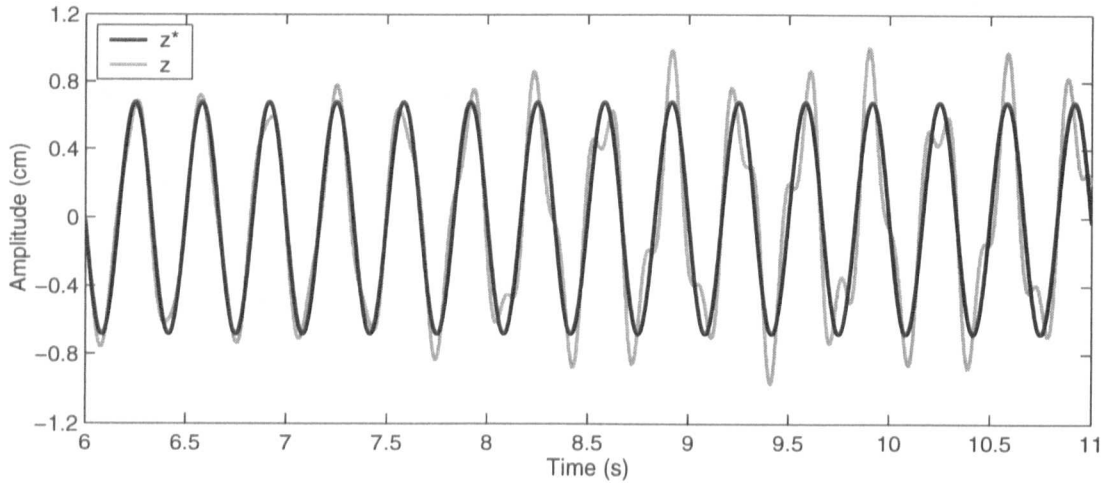


Figure 3.12: Transition to instability as the delay compensation is reduced on the experimental system, Forward Prediction $\rho \approx 2.6$ms.

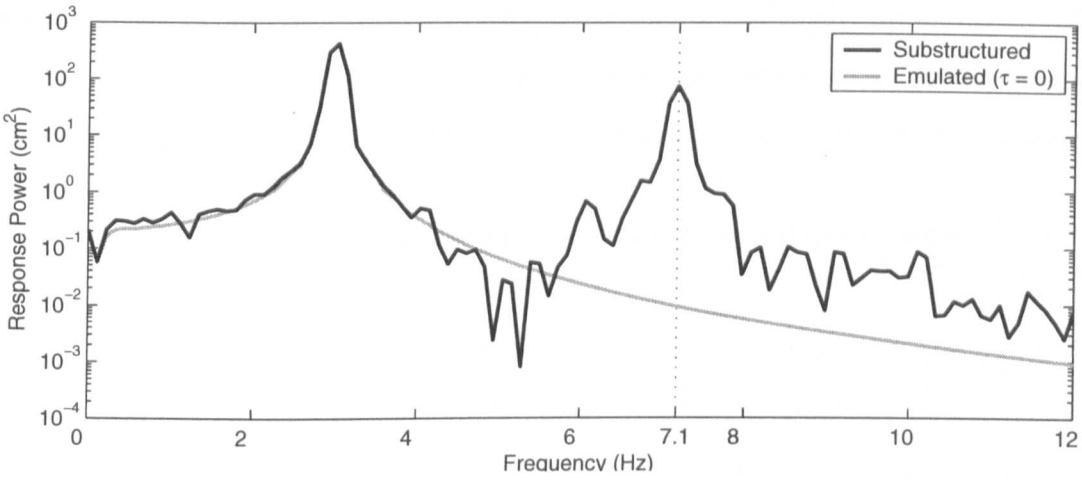


Figure 3.13: Frequency spectrum for the unstable experimental substructured response.

In addition to the any delay compensation error, the experimental tests were affected by approximately $5-7\%$ of noise on the feedback force signal, which makes it difficult to precisely determine the above values of τ_c and ω_I, as can be seen form Figure 3.14 unlike for the numerical simulations. However, despite this it can be seen that all the phase delay has been removed from the experimental system, the reason why

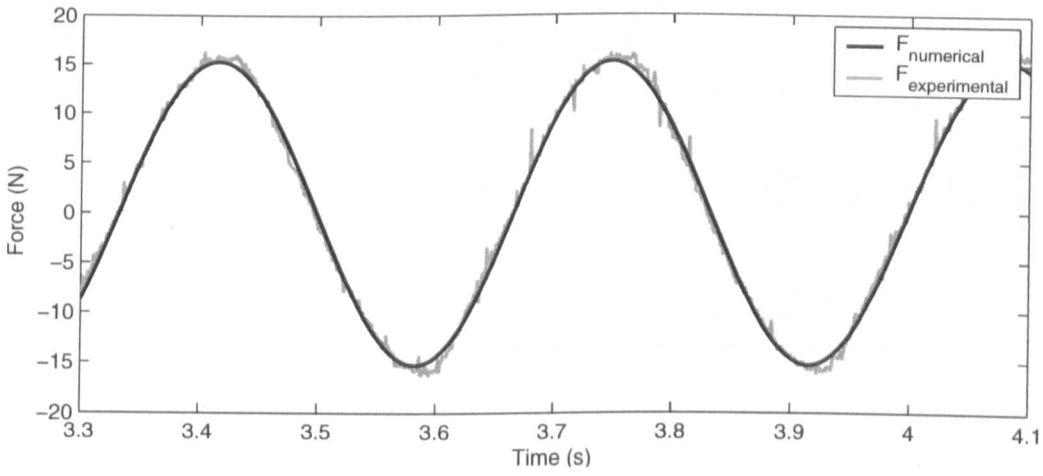


Figure 3.14: Comparison of the emulated force $F_{numerical}$ to that of the actual experimentally measured force $F_{experimental}$.

Eq. (3.4) is stable. This is a general issue for all experimental systems, nevertheless, the agreement with the two modelling techniques remains excellent.

3.4.1 Substructuring with very low damping

We note that the lower the damping in the system, the smaller the critical limit of stability and, therefore, the higher $\Re(\lambda)$ will be for a given response delay. Thus, the harder the controller must work to achieve the same results for a given delay. To

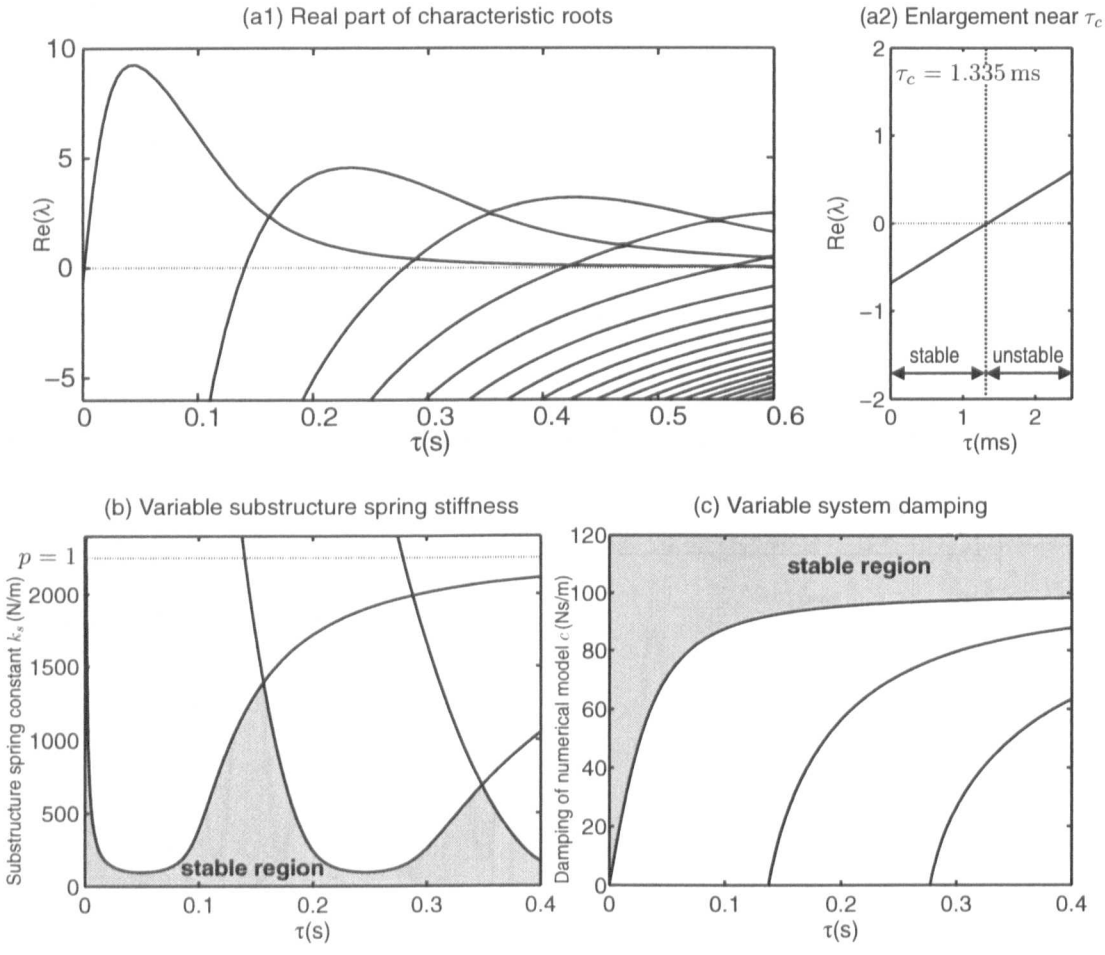


Figure 3.15: Repeat of Figure 3.2 for lower damping coefficient: (b) variable substructure spring stiffness k_s ($c = 3$Ns/m), and (c) variable system damping c ($k_s = 2250$N/m).

illustrate this we have repeated the experiment shown in Figure 3.11 with a similar test but with damping of only $\zeta = 0.0213$ ($c = 3$Ns/m) instead of the previous value of $\zeta = 0.1066$. This level of damping is more representative of the typical situation found in civil engineering applications.

Figure 3.15 shows the new stability bounds for the system using DDE-BIFTOOL as in § 3.2.3. The new critical delay is given in Figure 3.15(a2) as $\tau_c = 1.335$ms for the case of $p = 1$. Consequently, the asymptotic stability regions given in Figure 3.15(b) and (c) as much reduced. Figure 3.16 shows the results for a steady state experimental test. We see a similar result as the delay compensation scheme is still working effectively, however the margin to instability is now vastly reduced. The resulting substructured system is therefore far more volatile, such that the transition to instability is quicker (as the exponential growth is faster for the same magnitude of delay past the critical limit) and the effect of an external disturbance could more easily lead to instability and a catastrophic failure of the substructure. This leads us to the concept of *robustness* for a substructuring test which is discussed in detail

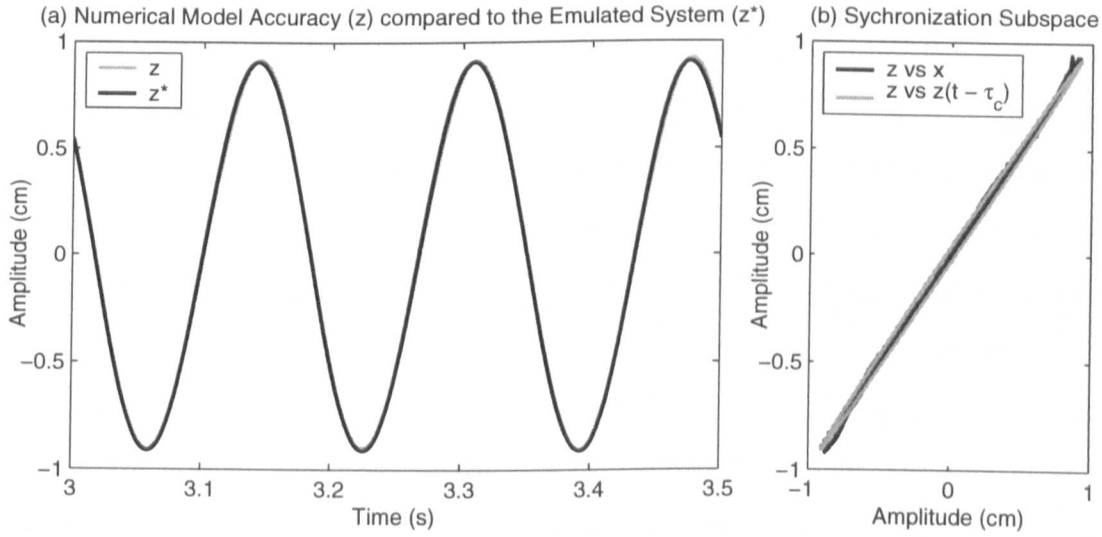


Figure 3.16: Experimental real-time dynamic substructure test with wall excitation of 3Hz and delay compensation of 9.4ms. Transfer system synchronisation is shown in (b) with limit of stability represented by the z vs $z(t - \tau_c)$ loop. τ_c=1.335ms.

in Chapter 5. In essence, improved robustness, increasing the margin to instability, comes at the cost of reducing dynamical accuracy when compared to the emulated system dynamics.

3.5 Conclusion

In this chapter we have discussed two approaches to representing the stability criteria for a given substructured system. We have been able to use well established techniques from two differing fields to determine the critical delay, τ_c, beyond which the substructured system is unstable. Specifically, using just a well-known simple linear example of a single mass-spring oscillator system all the dominant but complex characteristic effects that delays play in the dynamical accuracy and stability of the substructuring algorithm have been demonstrated. Furthermore this has been done using both analytical and numerical solutions. In order to validate these approach experimentally, the delay compensation scheme of Chapter 4 is utilised to good effect demonstrating the overall effectiveness of these modelling approaches.

The first method we employ here is to model the substructured system with *delay differential equations* (DDEs). The advantage of DDE modelling is that we can use powerful analytical and numerical methods to determine the stability of the DDE model and, hence, of the substructured system. The field of DDEs is well developed and as such we can use such tools as DDE-BIFTOOL to follow the Hopf bifurcations. However, there are some substructured systems that cannot be modelled in this way due the the characteristics of the DDE produced (3rd order DDE systems where the acceleration state is delayed). This is where the phase margin approach to calculating the relative and robust critical delay is brought to bear. The use of linear theory — and particularly the assumptions that the experimental substructure and transfer systems are approximated by linear transfer functions — would at first sight appear to be a serious limitation of this analysis. However, this technique can be applied to any generic substructured system, providing a good working

approximation of stability for complex systems.

Hybrid testing of lightly damped dynamical systems using numerical-experimental real-time substructuring is sensitive to both transfer system delay and uncertainty. It should be noted that the lower the damping in the system the greater the destabilizing the effect both the transfer system delay and uncertainty have on the substructuring algorithm. This leads to the *design* of a robust transfer system in Chapter 5.

Chapter 4

Delay compensation techniques

SUMMARY: This chapter presents the concept of using a delay compensation scheme, as an outer-loop control strategy, in order to achieve successful real-time substructure testing. The aim, regardless of the algorithm used, is to cancel out the inherent dynamic characteristics of the transfer system in order to negate the effect of the delay errors fed back into the substructuring algorithm.

4.1 Introduction

In the field of substructuring, the first time delay compensators were obtained by assuming that the dynamics of the transfer system may be approximated to a pure delay. For example, Horiuchi et al. [61] and Blakeborough et al. [1] proposed outer-loop forward prediction methods which use polynomial extrapolation to predict forward the numerical model displacement by a fixed number of time-steps. Darby et al. [73] relaxed the assumption of a pure delay by developing a forward prediction method that varied the amount of delay compensation, based on the error between the actuator displacement and the desired numerical model displacement. This method was extended by Wallace et al. [75] who developed an adaptive forward prediction algorithm that used variable polynomial coefficients such that non-integer

multiples of the previous time step could be predicted and also incorporated an amplitude correction algorithm (discussed in detail in § 4.3.6). The use of a Smith predictor has also been proposed as a suitable delay compensator Sivaselvan et al. [68].

Lag compensation via an experimental transfer function estimation of the combined inner-loop controller and actuator dynamics has been proposed by Gawthrop et al. [76] (discussed in detail in § 4.4.1) and Sivaselvan et al. [68]. The proposed outer-loop controllers compensate for unwanted dynamics by applying the inverse of the transfer function estimation. Model reference adaptive control has also been suggested as an outer-loop strategy by Wagg and Stoten [67], Neild et al. [77] and Lim et al. [78] which demonstrated how lag compensation can be achieved via this approach.

4.2 Application to delay compensation

The value of being able to compensate for the transfer system dynamics can be seen from Figure 4.1. If the *inner-loop* proprietary controller is designed correctly (see § 2.3.4), then the *response* of the transfer system x to the *demand* of the numerical model z will be accurate and have low uncertainty apart from the inevitable time delay τ (expected in any dynamical system under proprietary control). This delay may or may not be frequency dependent depending on the type of transfer system being used. If we use an additional *outer-loop* controller which creates a new reference signal, z', in front of original numerical model signal and then use this as the *new* demand for the inner-loop controller, the magnitude of the transfer system delay can be reduced. As the phase advance of this new reference signal increases in magnitude the synchronisation error e_2 (the local substructuring error, § 2.6) decreases until it matches the actual delay of the transfer system τ. At which point, complete compensation is said to have been achieved as the transfer system dynamics will be have been nominally *cancelled* out.

From Chapter 3 it is clear that the magnitude of delay of the transfer system τ

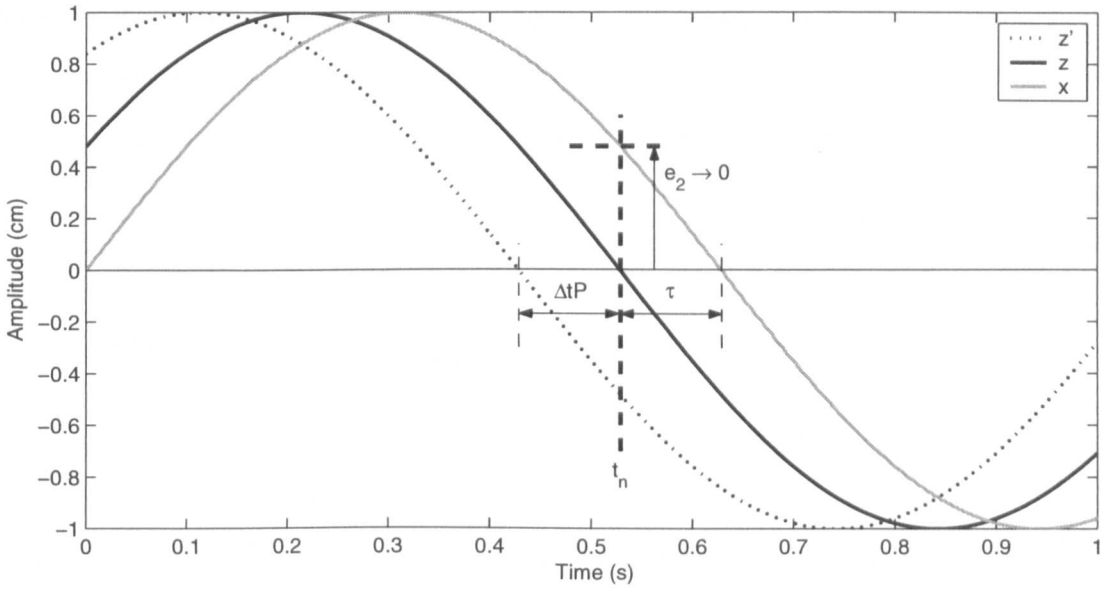


Figure 4.1: Delay compensation for the transfer system dynamics by creating a new reference signal z'

has a significant effect on the dynamics of the numerical model as the correct force vector $F(t)$ must be added into the numerical model at the correct time. In fact, Proposition 1 tells us that the accuracy of the numerical model, e_1, is directly coupled to the synchronisation error of the transfer system, e_2. If $\tau > \tau_c$ then the substructuring algorithm will be unstable and is the typical case in substructuring when using the inner-loop controller in isolation. As the magnitude of delay compensation is increased τ will decrease until $0 < \tau < \tau_c$ at which point the substructuring algorithm will be stable but the dynamical accuracy of the numerical model will be compromised, as the transfer system delay is still adding negative damping into the system. When $\tau = 0$ then the synchronisation error $e_2 = 0$ and thus the numerical model dynamics will now replicate those of the emulated system, z^*, as $e_1 = 0$. However, in practice the synchronisation error can never be exactly equal to zero in a real substructuring test (due to the effects of noise and experimental errors) so the practical interpretation of Proposition 1 is that as $e_2 \rightarrow 0$ the substructured model more closely replicates the dynamics of the emulated system.

It is possible to further increase the level of compensation such the transfer system would lead in front of the numerical model, such that $\tau < 0$. This situation would result in adding extra damping into the substructured system and is discussed further in § 5.5 in relation to creating a robust substructuring algorithm.

4.3 Delay compensation via polynomial extrapolation

Figure 4.2 shows an example sinusoid reference signal of 10Hz (shown by the solid grey line) which is to be predicted forward. A section of data must be taken to act as control points for the fitted polynomial curve, here a buffer of 20 data points $n = 20$ starting at time $t = 1$ has been stored. For example, if we wish to predict forward 18 time steps ($P = 18$ and Δt is the sample time step size and equal to 1ms in this case) Figure 4.2 shows the differing accuracies obtained by the various order N of polynomial fitted curves. From this example, we see that both the 8th and 10th order curves have the highest degree of accuracy, whereas the 4th order curve loses coherence more quickly. However, the higher the order of prediction the

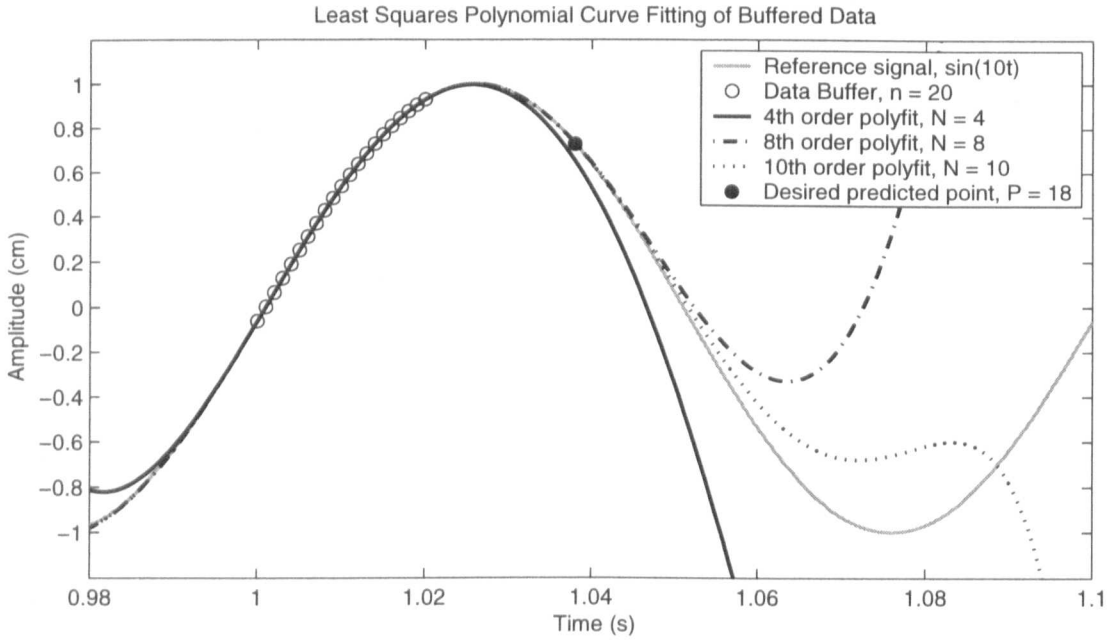


Figure 4.2: Least squares N^{th} order polynomial curve fitting of buffered data.

more computationally intensive the calculation and the more inherently unstable the predicted signal z' will become.

To achieve optimal delay compensation the desired amount of time for which the numerical model should be predicted forward is equal to $\tau = \triangle tP$.

4.3.1 Least squares polynomial fitting

One mathematical procedure for finding the best fitting curve to a given set of data points is by minimising the sum of the squares of the offsets (known as residuals) of the points from the curve, an example is shown in Figure 4.3 [100]. The sum of the squares is used instead of the offset absolute values because this allows the residuals to be treated as a continuous differentiable quantity. However, because squares of the offsets are used, outlying points can have a disproportionate effect on the fit, a property which may or may not be desirable depending on the problem. In practice, the vertical offsets from a line are almost always minimised instead of the perpendicular offsets. This provides a fitting function which is independent for

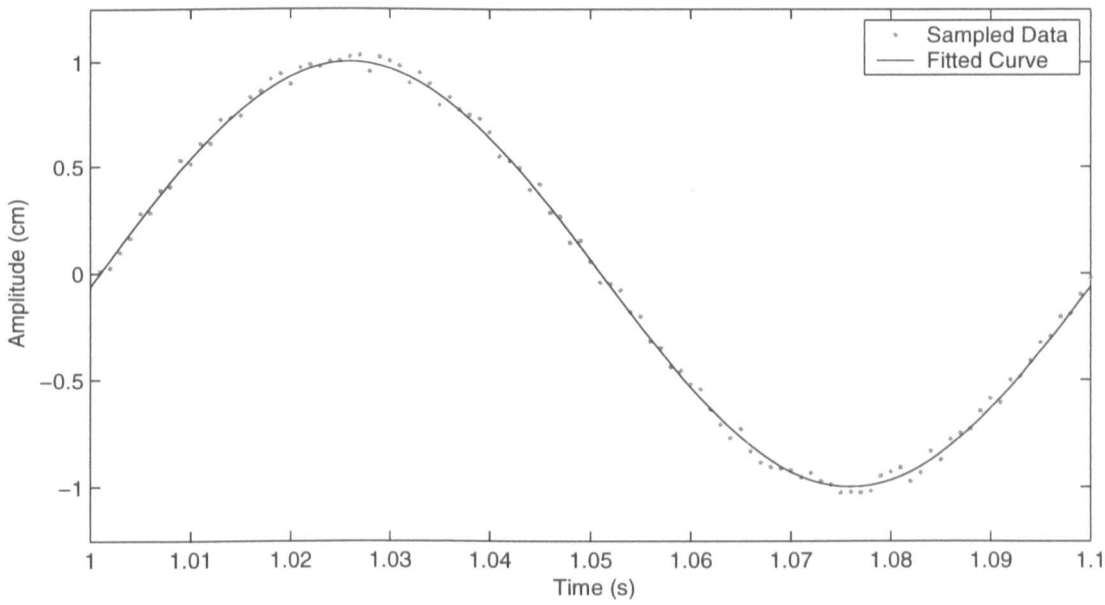


Figure 4.3: Generalised curve fitting of a 10Hz sinusoid to discrete data points.

each variable, allows uncertainties of the data points along the x- and y-axes to be incorporated simply, and also provides a much simpler analytic form for the fitting parameters than would be obtained using a fit based on perpendicular offsets.

The linear least squares fitting technique is the simplest and most commonly applied form of linear regression and provides a solution to the problem of finding the best fitting straight line through a set of points. However, due to the dynamics of the numerical models in our systems we move from a best-fit line to a best-fit polynomial. A polynomial is a mathematical expression involving a sum of powers in one or more variables multiplied by coefficients. A polynomial in x of order N with coefficients a_i (where $i = 0, \ldots, N$) is given by,

$$y = a_0 + a_1 x + \ldots + a_N x^N. \tag{4.1}$$

Using a standard least squares polynomial derivation we can deduce an equation to describe the coefficients of the curve [100]. From Eq. (4.1) the square of the residual is given by

$$R^2 \equiv \sum_{i=1}^{n} [y_i - (a_0 + a_1 x_i + \ldots + a_N x_i^N)]^2, \tag{4.2}$$

where n is the number of data points. The partial derivatives of Eq. (4.2) can then be obtained, these are given by

$$\frac{\partial(R^2)}{\partial a_0} = -2 \sum_{i=1}^{n} [y_i - (a_0 + a_1 x_i + \ldots + a_N x_i^N)] = 0, \tag{4.3}$$

$$\frac{\partial(R^2)}{\partial a_1} = -2 \sum_{i=1}^{n} [y_i - (a_0 + a_1 x_i + \ldots + a_N x_i^N)] x_i = 0, \tag{4.4}$$

$$\frac{\partial(R^2)}{\partial a_N} = -2 \sum_{i=1}^{n} [y_i - (a_0 + a_1 x_i + \ldots + a_N x_i^N)] x_i^N = 0. \tag{4.5}$$

We can rewrite these equations in the following form

$$a_0 n + a_1 \sum_{i=1}^{n} x_i + \ldots + a_N \sum_{i=1}^{n} x_i^N = \sum_{i=1}^{n} y_i, \tag{4.6}$$

$$a_0 \sum_{i=1}^{n} x_i + a_1 \sum_{i=1}^{n} x_i^2 + \ldots + a_N \sum_{i=1}^{n} x_i^{N+1} = \sum_{i=1}^{n} x_i y_i, \tag{4.7}$$

$$a_0 \sum_{i=1}^{n} x_i^N + a_1 \sum_{i=1}^{n} x_i^{N+1} + \ldots + a_N \sum_{i=1}^{n} x_i^{2N} = \sum_{i=1}^{n} x_i^N y_i, \tag{4.8}$$

which can in tern be represented in matrix format as follows

$$\begin{bmatrix} n & \sum_{i=1}^n x_i & \cdots & \sum_{i=1}^n x_i^N \\ \sum_{i=1}^n x_i & \sum_{i=1}^n x_i^2 & \cdots & \sum_{i=1}^n x_i^{N+1} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^n x_i^N & \sum_{i=1}^n x_i^{N+1} & \cdots & \sum_{i=1}^n x_i^{2N} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_N \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^n y_i \\ \sum_{i=1}^n x_i y_i \\ \vdots \\ \sum_{i=1}^n x_i^N y_i \end{bmatrix}. \tag{4.9}$$

Eq. (4.9) is the equation for the least squares polynomial fit. The first matrix of this equation is called a Vandermonde matrix [100]. We can also deduce the matrix for the least squares fit by writing

$$\begin{bmatrix} 1 & x_1 & \cdots & x_1^N \\ 1 & x_2 & \cdots & x_2^N \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \cdots & x_n^N \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_N \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}. \tag{4.10}$$

Premultiplying both sides by the transpose of the first matrix then gives

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ \vdots & \vdots & \ddots & \vdots \\ x_1^N & x_2^N & \cdots & x_n^N \end{bmatrix} \begin{bmatrix} 1 & x_1 & \cdots & x_1^N \\ 1 & x_2 & \cdots & x_2^N \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \cdots & x_n^N \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_N \end{bmatrix} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ \vdots & \vdots & \ddots & \vdots \\ x_1^N & x_2^N & \cdots & x_n^N \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \tag{4.11}$$

which, if multiplied out, gives a solution equal to Eq. (4.9). Therefore, given n data points (x_i, y_i) with fitted polynomial coefficients $(a_0, \ldots, a_n)$ we can write

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^N \\ 1 & x_2 & x_2^2 & \cdots & x_2^N \\ \vdots & \vdots & \ddots & \vdots & \\ 1 & x_n & x_n^2 & \cdots & x_n^N \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_N \end{bmatrix}. \tag{4.12}$$

Therefore, in matrix notation, the equation for a polynomial fit is given by

$$y = Xa, \tag{4.13}$$

which can be solved by premultiplying by the matrix transpose X^T,

$$X^T y = X^T X a. \tag{4.14}$$

We can solve the matrix equation numerically, or it can be inverted directly to give the solution vector given by

$$a = (X^T X)^{-1} X^T y, \tag{4.15}$$

where setting $n = 1$ will give the linear solution.

4.3.2 Single step forward prediction

Delay compensation by polynomial extrapolation is not a new concept, single time step prediction techniques have already been proposed in relation to substructuring by Horiuchi et al. [61] and Darby et al. [82]. These algorithms are based on using predefined coefficients, $\bar{a}_i$, for an N^{th} order polynomial fit of n number of control points following the equation,

$$z' = \sum_{i=0}^{N} \bar{a}_i \, z_i, \tag{4.16}$$

where z_0 is the present calculated numerical model displacement and z_i are the previously calculated displacements at $\Delta t \times i$ units of time ago. Figure 4.4 shows the forward predicted point z' being obtained by extrapolating the polynomial function over the present displacement z_0 and N previous calculated values, thus making the number of control points used, $n = (N + 1)$.

For this 2^{nd} order polynomial fit we attain the following constants $\bar{a}_0 = 3$, $\bar{a}_1 = -3$ and $\bar{a}_2 = 1$ [61]. Note that we can only predict one whole time step forward and the correct number of control points n must used for the polynomial function (to ensure that the X matrix of Eq. (4.13) is square) otherwise the polynomial fit will not hold. To predict further ahead than one time step we can simply apply Eq. (4.16) more than once [82]. For example to predict two time steps forward we see that if

$$z'^{(\Delta t)} = 3z_0 - 3z_1 + z_2, \tag{4.17}$$

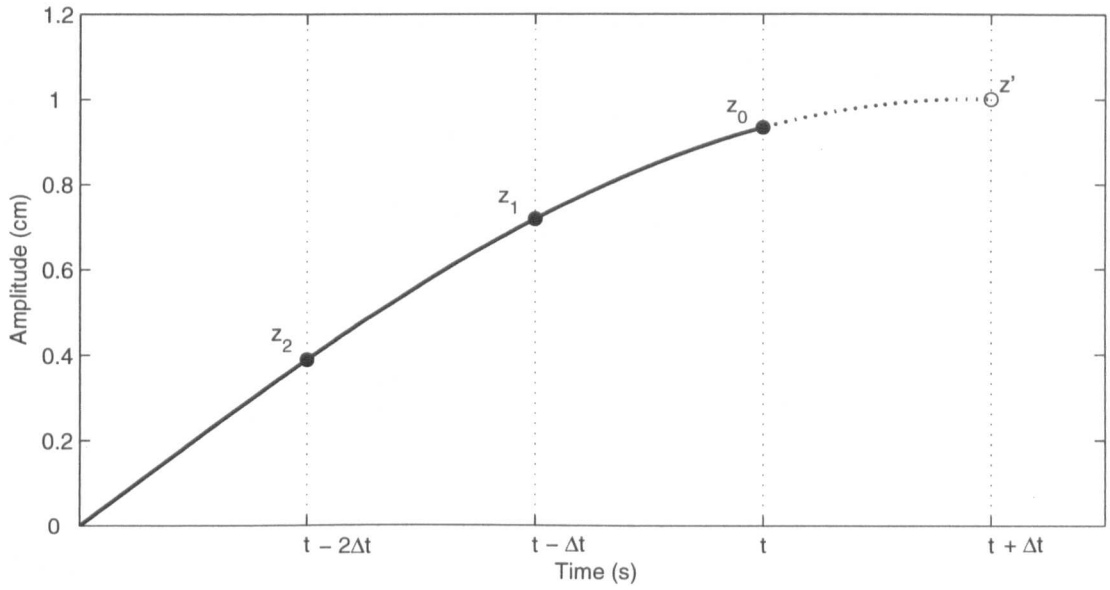


Figure 4.4: Single time step prediction of $n = 3$ control points for a $N = 2$ order polynomial fit.

then,

$$\begin{aligned} z'^{(2\Delta t)} &= 3z'^{(\Delta t)} - 3z_0 + z_1, \\ &= 6z_0 - 8z_1 + 3z_2. \end{aligned} \tag{4.18}$$

Note that we are still restricted to predicting in whole time steps unless an additional interpolation is carried out between the two points [82].

However, if we use Eq. (4.15) to solve the $(X^T X)^{-1} X^T$ matrix components numerically first, we can deduce a more generalised forward prediction algorithm where multiples and fractions of one time step can be predicted in one iteration and we are no longer constrained to $n = (N+1)$ number of control points. Increasing n can help to smooth noise out of the numerical model and thus from the control signal. We define the size of the X matrix by choosing values for the number of control points n and the order of the polynomial fit N. For the case above, $n = 3$ and

$N = 2$, thus rewriting the general Eq. (4.13) for this specific case gives us,

$$\begin{bmatrix} z_0 \\ z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} 1 & t_0 & t_0^2 \\ 1 & t_1 & t_1^2 \\ 1 & t_2 & t_2^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}, \tag{4.19}$$

where, t_i is the current simulation time for each value of z_i. Therefore, for a sample time step size Δt as shown in Figure 4.4,

$$\begin{bmatrix} z_0 \\ z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & -\Delta t & \Delta t^2 \\ 1 & -2\Delta t & 4\Delta t^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}. \tag{4.20}$$

The predicted point z' is given by an adaptation of Eq. (4.13),

$$z' = X_P \, a, \tag{4.21}$$

where, X_P is the forward prediction vector and given by,

$$X_P = [1 \quad P\Delta t \quad \ldots \quad P^N \Delta t^N], \tag{4.22}$$

and P is the number of time steps to be predicted forward (which does not have to be an integer multiple of Δt). As X is a square matrix in this case, $(X^T X)^{-1} X^T = X^{-1}$, therefore,

$$z' = X_P X^{-1} z. \tag{4.23}$$

Evaluating $X_P X^{-1}$ gives an expression which is independent of the sample time step size Δt. Substituting into Eq. (4.23) for the case when $P = 1$ we see that

$$z'^{(\Delta t)} = 3z_0 - 3z_1 + z_2, \tag{4.24}$$

thus matching the coefficients of the one step method in Eq. (4.17). Substituting $P = 2$ we see that

$$z'^{(2\Delta t)} = 6z_0 - 8z_1 + 3z_2, \tag{4.25}$$

matching the coefficients found in Eq. (4.18) but in a single operation. Therefore, the coefficients $\bar{a}_i$ from Eq. (4.16) are actually the pre-multiplication of the forward prediction vector X_P for the special case of $P = 1$, such that

$$\bar{a} = X_P[(X^T X)^{-1} X^T]. \tag{4.26}$$

In this way, the coefficients $\bar{a}$ are predefined and thus fix the level of forward prediction obtained, whereas the coefficients of the polynomial function a are calculated each time step and therefore allow variable degrees of forward prediction to be achieved by altering the value of P online.

4.3.3 Online forward prediction using variable coefficients

To achieve online forward prediction a buffer of n data points of the numerical model displacement z are stored and updated by a buffer overlap of $(n-1)$ each time step. These are then fed into a least squares polynomial fitting sub routine which uses the set of control points $[z_{n-1}, z_{n-2}, \ldots, z_0]$ to calculate the N^{th} order polynomial fit and therefore find the coefficient vector a from Eq. (4.15) for that time step. The current coefficients are then fed into a reconstruction algorithm that calculates the predicted point z' according to Eq. (4.21) using the forward prediction vector X_P, such that

$$z' = k_a \sum_{i=0}^{N} (a_i P^i). \tag{4.27}$$

When using linear control, an additional source of error is the amplitude accuracy. Typically, as the excitation frequency is increased the level of amplitude accuracy of the transfer system reduces. Thus, the forward predicted point z' is then multiplied by a gain k_a to remove this error and increase the synchronisation accuracy.

An example of the layout for an online forward prediction model is shown in Figure 4.5. Note that due to the structure of the Matlab *Polyfit* subroutine, the points must be re-indexed by a factor of $(n-1)$ in order to calculate the predicted point z' correctly. Additionally, the buffer must be completely filled at least once before the prediction process has been fully initialised. This issue can be resolved in a number of ways; by zero padding, which consists of extending the signal back in time with zeros, of length m (where, $m > n$), or by activating the algorithm only after the buffer has been filled. Note that the numerical model reference signal, z, does not have to be a sine wave, but must be some form of smooth signal. A sharp signal

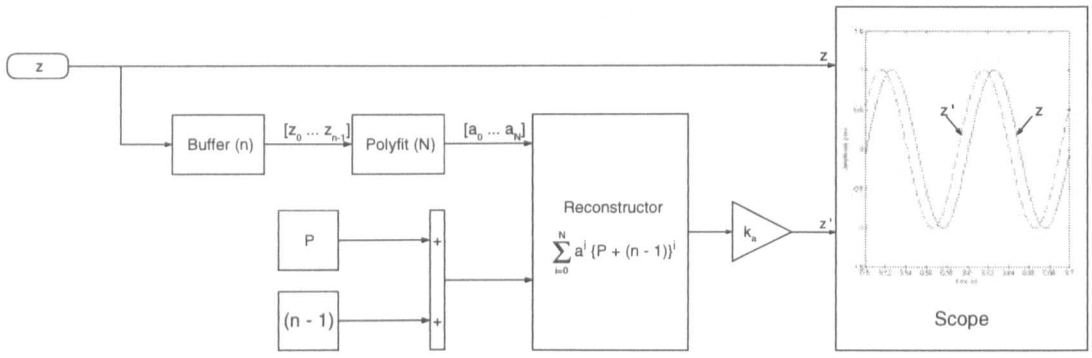


Figure 4.5: Basic online forward prediction model structure.

cannot be extrapolated in this manner.

We define the predictor frequency, $f_{predictor}$, as the maximum frequency at which we can achieve any magnitude of forward prediction. Figure 4.6 shows the relationship between the maximum number of time steps that can be predicted forward, P_{max}, at a number of different frequencies up to $f_{predictor}$ for set algorithm parameters of $N = 8$ and $n = 20$. An inverse relationship is observed, with $f_{predictor} \approx 57$Hz in this case. We cannot explicitly calculate $f_{predictor}$ as it depends the algorithm parameters of Δt, N and n. However, we can use this inverse relationship to extrapolate the predictor frequency for a given set of algorithm parameters with only a few experimental data points. We can now examine the effect each of the algorithm parameters has on $f_{predictor}$.

The smaller the sample time step size, Δt, the greater the relative range of the prediction algorithm, regardless of the other algorithm parameters. This is because although the algorithm allows more time steps to be predicted the actual distance in time that must be predicted forward decreases as Δt is reduced, as $\tau = \Delta t P$. However, for experimental purposes the forward prediction algorithm is restricted by the real-time requirements of the hybrid testing methodology, making the choice of Δt for a specific model straight forward. Typically, Δt is made as small as possible but still allowing the full model computation to be carried out within the hard time

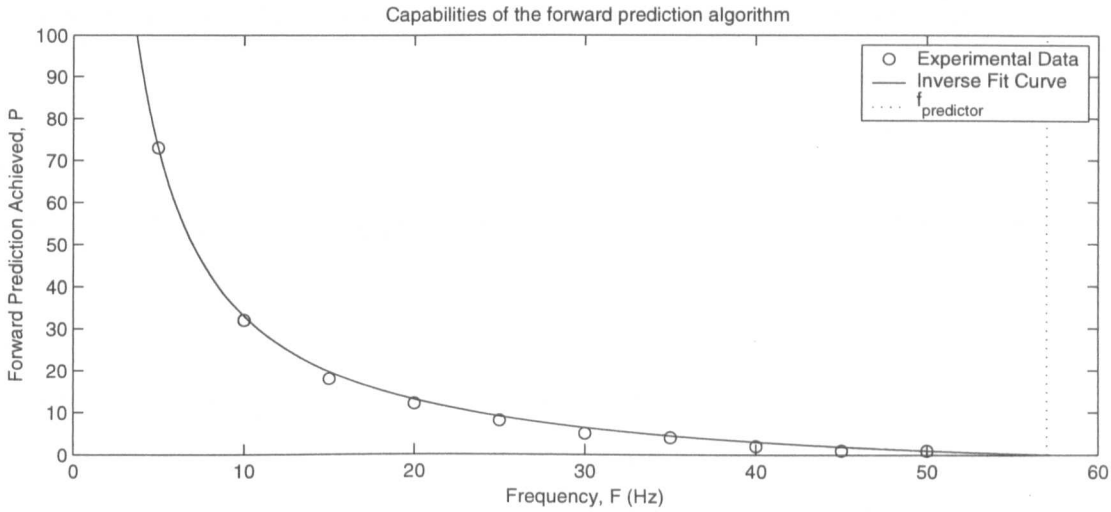


Figure 4.6: Magnitude of P_{max} as the input frequency f is increased. Algorithm parameters: $N = 8$, $n = 20$ and $\Delta t = 0.001$.

constraints.

The higher the order of the polynomial N, the greater its ability to fit to the input reference signal. The more complicated the input reference signal the higher the order of the polynomial fit needs to be, especially if the forward prediction algorithm must cope with complex harmonic sinusoids. Additionally, the higher N the greater the ability of the polynomial fit to achieve coherence at greater values of P, the amount of time steps predicted forward. However, there is a limit to how high the order of the polynomial can be as simply increasing the order will not always increase coherence. Polynomial extrapolation is inherently unstable, meaning that high levels of prediction with high order polynomial curves can lead to instability rather than just a loss in coherence. We therefore have a trade off between choosing potential high order accuracy with low order stability. It is important to note that as soon as noisy signal if fed into the algorithm the instability becomes increasingly apparent. Additionally, it is also important to note in a real-time experimental situation there is a computational overhead with using a higher order polynomials. Therefore, as small as possible a polynomial fit N should be chosen, only increasing

the order if coherence cannot be achieved over the desired range up to the predictor frequency $f_{predictor}$.

Keeping the order of the polynomial N constant (let $N = 8$ for example), we can now observe the effect of varying the number of control points n, the size of the data buffer. Figure 4.7 shows the linear relationship between n and P_{max} up until the point where the number of control points is too low to create an accurate polynomial fit. This minimum limit is set at $n = (N + 1)$ such that Eq. (4.13) can be solved. Comparing the results for the input reference of 5 Hz and 10 Hz we see a constant gradient, just a changing value for the y-intercept constant. Note that the constant of proportionality here is less than one. This means that the smaller n is (up to the cut off limit) the greater distance in time we can predict forward. Therefore to achieve the peak performance from the forward prediction algorithm the number of control points should be set as low as possible. However, as the algorithm also acts as a smoothing filter, by increasing the number of control points the effect of noise on the signal to be predicted can be significantly reduced. This is especially helpful when using higher order polynomials as they are inherently more unstable. As a

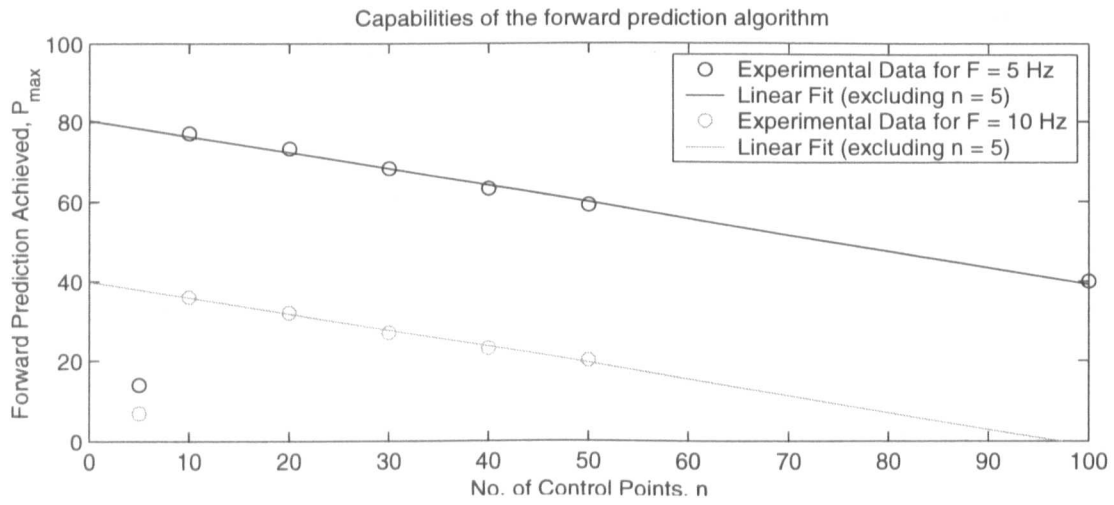


Figure 4.7: Increased prediction capabilities by reducing the number of control points n up to the minimum limit of $n = (N + 1)$, where $N = 8$.

guideline, a suitable range of control points is given by

$$(N+1) <= n <= 4N. \tag{4.28}$$

4.3.4 Method for setting parameter values

A general routine for setting the predictor parameter values is as follows:

1. Find the smallest step size Δt for which the model can run in real-time.

2. Estimate the maximum frequency of the test, f_{test}, using a safety factor of your choice.

3. Perform a system identification of the plant to be controlled to establish the likely delay, τ, that will occur over the frequency range of interest, giving a value for P_{max}.

4. Chose the order of the polynomial fitted curve by inspecting the input reference signal. The more complicated the signal, the higher the order N of the polynomial must be. However, the higher N is the more control points n that are required to create it and the more likely the algorithm will be unstable at high values of P. A practical limit for experimental purposes is $N <= 8$.

5. Chose the number of control points n within the range of Eq. (4.28). The noisier the input reference signal, the larger n should be.

6. Perform a coherence test to ensure that the algorithm can effectively predict to the desired requirements and that algorithm is stable. Calculate P_{max} for a small number of input frequencies and then extrapolate $f_{predictor}$.

7. If $f_{predictor}$ is smaller than f_{test}, or P_{max} cannot be achieved with the current values for Δt, N and n, then the values for N or n must be changed, or, as a last resort, the model must be streamlined to reduce Δt as this will increase the forward prediction capabilities of the algorithm.

It is important to note that this process does not have to be carried out every time the prediction algorithm is used. For example, in conjunction with the experimental tests of the case studies discussed in this thesis it was found that a good general predictor, that will suffice for almost all situations, is where the number of control points is $n = 10$ and the order of the fitted polynomial curve is $N = 4$.

Additionally, this forward prediction algorithm has been designed specifically for a sinusoidal input due to the nature of the numerical model used in the substructured system. If a stiffer signal was required to be predicted forward the order of the polynomial curve would have to be significantly reduced to account for sharp edges.

4.3.5 Constant forward prediction

The basic forward prediction algorithm can be used to effectively remove the transfer system delay, however, both the magnitude of the forward prediction, P, and the amplitude gain, k_a, must be specifically *tuned* for each different excitation condition. When used as an outer-loop strategy in conjunction with the inner-loop proprietary controller of the substructuring algorithm, it is simply a feed-forward controller.

To show the effectiveness of the basic forward prediction algorithm, we perform a system identification test for each transfer system (from the MDOF case study, refer to § 2.3.2) at a frequency of excitation of $r_{1,2} = 4.5Hz$. Figure 4.8 shows the subspace plots for both transfer systems for such a test. We can see from Figures 4.8 (a2) and (b2) a significant improvement in the level of synchronisation between each respective numerical model and the transfer system response for the case when the algorithm is activated compared to when a standard linear controller is used in isolation, Figures 4.8 (a1) and (b1). The *tunable* parameter values are set for each transfer system to achieve optimum synchronisation for this excitation condition, the exact values are shown in the figure caption. Although the excitation demands are identical and the actuators are of the same type, we can see that the transfer system dynamics vary considerably as shown by the differing subspace patterns in

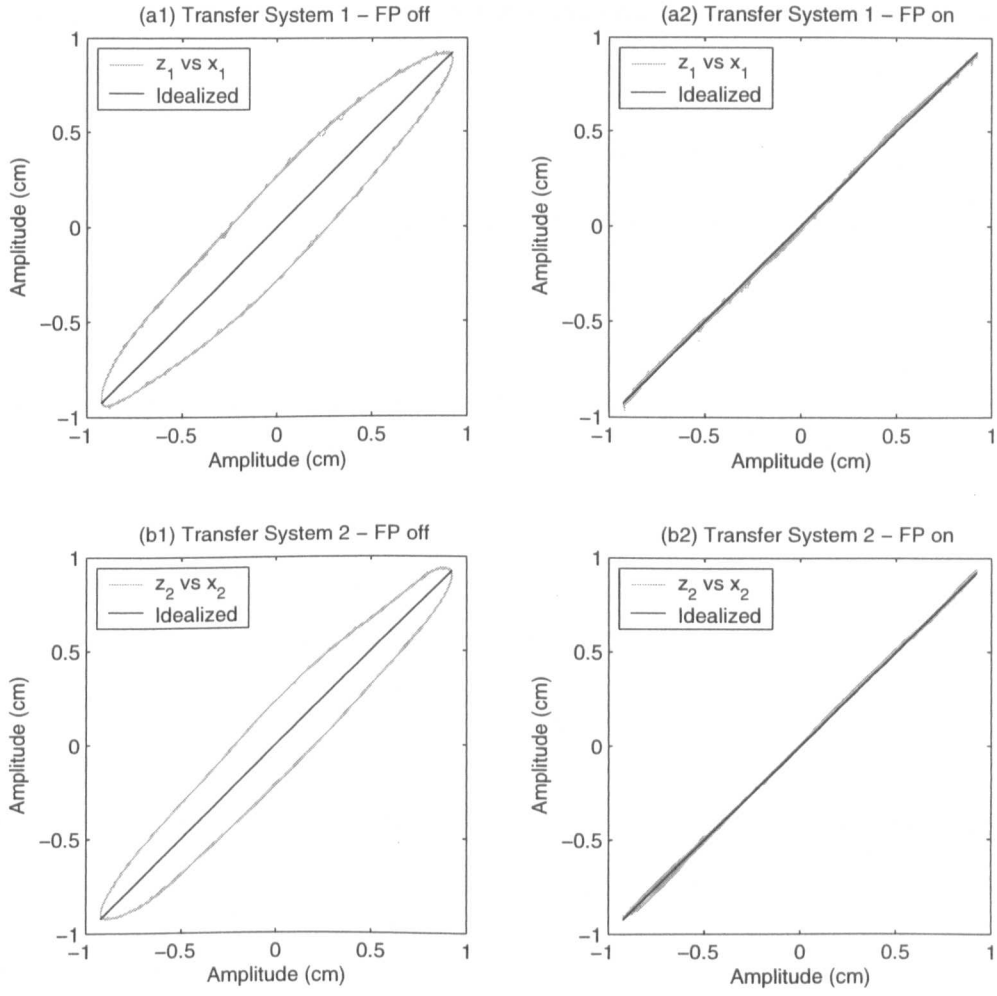


Figure 4.8: Synchronisation subplots for a system identification test for each transfer system at an excitation of $z_{1,2} = 4.5Hz$. Algorithm parameters: $\Delta t = 0.001$, $N = 4$, $n = 10$. *Tunable* parameters: $P_1 = 9.4$, $P_2 = 8.6$, $k_{a1} = 0.99$, $k_{a2} = 0.98$.

Figures 4.8 (a1) and (b1), and in the values of the tunable parameters. It can be seen that the delay error is by far the dominant factor in the synchronisation error. This highlights the need for the dynamics of the transfer systems to be decoupled such that their specific mechanical characteristics can be dealt with independently. It should be noted that although the dynamic characteristics of the transfer systems used is best characterized by a pure delay, the exact value can vary as it is only an approximation along with the amplitude accuracy which has a tendency to overshoot at high frequency due to the fact that the actuators are velocity controlled.

4.3.6 Adaptive forward prediction (AFP)

In order to remove the need for tuning the prediction parameters and to allow the algorithm to achieve high levels of synchronisation for frequency dependent and transient plant conditions we must close the control loop and use the feedback dynamics of the transfer system. In combination with the existing linear control present in the substructuring algorithm this model structure now represents an error driven adaptive feedback controller [101]. Figure 4.9 shows schematically how the basic forward prediction algorithm, shown in Figure 4.5, can be altered to achieve self tuning. We cannot explicitly measure the transfer system delay τ as we only have data for the current time step, thus we only know the synchronisation error e_2 at any single point in time. Therefore, to achieve complete delay compensation we can indirectly force $\tau \rightarrow 0$ by explicitly using a measure of the synchronisation error e_2. An alternative technique which uses this feedback error to achieve adaptive compensation is presented by Darby et al. [73].

As before, the current coefficients are fed into a reconstruction algorithm that calculates the predicted point z' but now the magnitude of the forward prediction

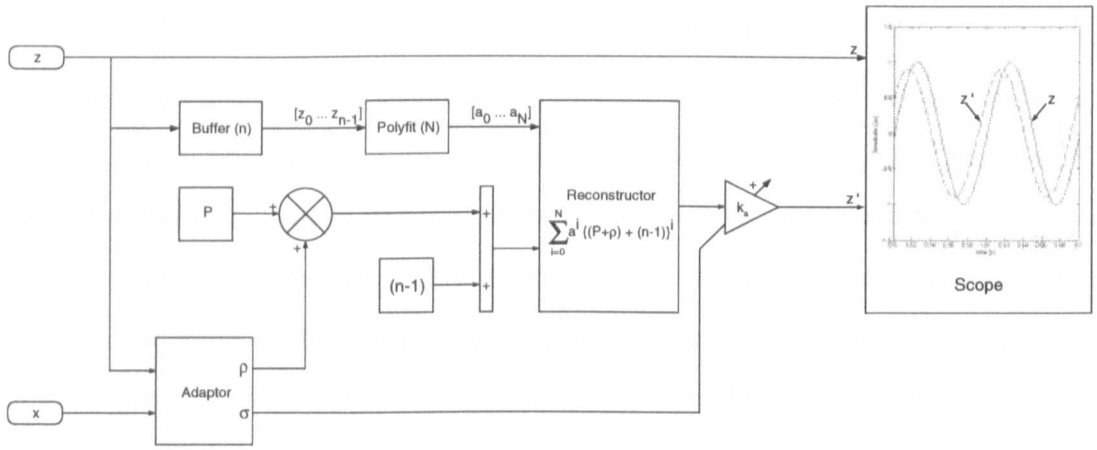


Figure 4.9: Adaptive forward prediction (AFP) model structure.

τ is now governed by,

$$\tau = \Delta t(P + \rho), \tag{4.29}$$

where, Δt is the sample time step size, P is the *fixed* initial number of time steps to be forward predicted and ρ is the *adaptive* number of time steps to be forward predicted. Likewise, the amplitude accuracy is now governed by,

$$z' = z'(k_a + \sigma), \tag{4.30}$$

where, k_a is the *fixed* amplitude gain and σ is an *adaptive* amplitude variable which together control the amount of the predicted reference signal to be used.

Setting $P = 0$ and $k_a = 1$ will bring about zero initial conditions. The delay compensation can then be completely achieved by the adaptive parameter ρ and the amplitude error completely removed by σ. Thus, we can use this new adaptive algorithm when we have no knowledge of the plant dynamics and when there is transient or frequency dependent plant behaviour. However, it is not always desirable to start from zero initial conditions, in fact in many cases (and mainly in earthquake engineering) it is important to start the test with the AFP algorithm in a state near to optimal adaptation. These optimal values can be estimated by observing the steady state adaptive values of an uncoupled system identification test for the specific transfer system. This would then allow the initial transient phase of the test to be avoided.

A schematic of the Adaptor block of Figure 4.9 is shown in Figure 4.10. The adaption algorithm works off four triggered states, $\varphi_{1,\ldots,4}$, which have a null value until their individual trigger conditions are met. Trigger states, φ_1 and φ_2, are activated on the condition of sign change of the numerical model displacement, when z has zero amplitude. The first state represents a rising edge, z changing from negative to positive, and the second a falling edge, z changing from positive to negative. The other two trigger states are similar except activated on the condition of sign change of the numerical model velocity, $\dot{z}$, with φ_3 representing a rising edge and φ_4 representing a falling edge. Effectively these two conditions give the time at which

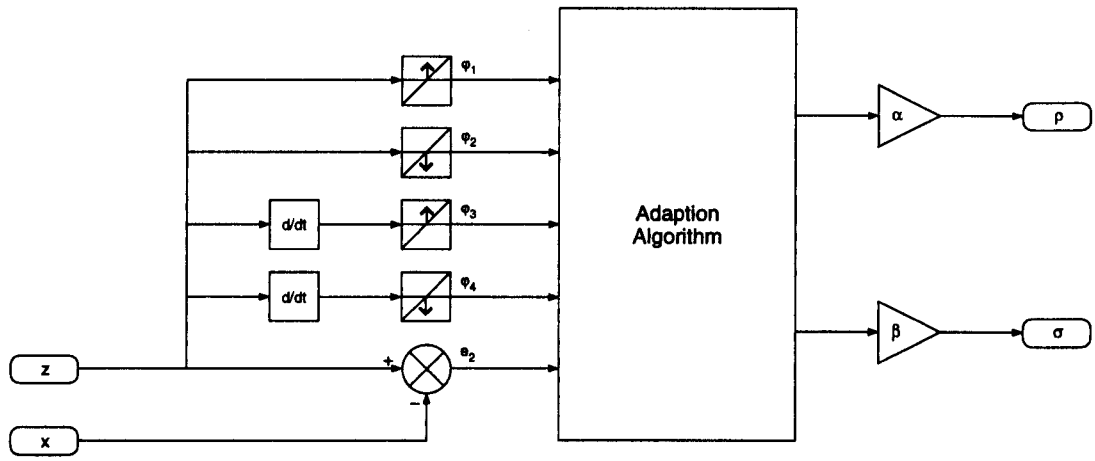


Figure 4.10: Adaptor model structure, where $\varphi_{1,\ldots,4}$ are trigger states of the reference signal and e_2 is the synchronisation error.

the maxima φ_3 and minima φ_4 of the numerical model displacement occurs.

The adaptive forward prediction parameter ρ is calculated when either of the first two triggered states $\varphi_{1,2}$ are met. This allows the delay of the transfer system response to be observed independently from any amplitude error. The value of ρ is given by,

$$\rho_{n+1} = \rho_n \pm \alpha e_{2,n}^{\gamma} , \tag{4.31}$$

where, α is an adaptive gain parameter for the magnitude of forward prediction and γ sets the convergence curve and must be greater than or equal to 1. Note that the $\pm$ relates to whether the signal is a rising or falling edge. Eq. (4.31) shows that when the synchronisation error e_2 is zero, $\rho_{n+1} = \rho_n$ thus ρ retains its previous value, indicating that full delay compensation τ has been achieved.

Similarly, the adaptive amplitude parameter is calculated when either of the second two triggered states $\varphi_{3,4}$ are met. This allows the peak of the numerical model to be compared to that of the transfer system once delay compensation has occurred. The value of σ is given by,

$$\sigma_{n+1} = \sigma_n \pm \beta e_{2,n}^{\gamma} , \tag{4.32}$$

where, β is an adaptive gain parameter for the amplitude accuracy. Note that until full delay compensation has been achieved the parameter σ will give an under estimation of the amplitude error. However, as the delay is by far the dominant factor in the compensation algorithm this condition does not effect the performance of the controller.

We see that by setting both adaptive parameters α and β to zero, the adaption algorithm can be turned 'off' resulting in the basic feed-forward controller, assuming both ρ_0 and σ_0 equal zero. Note that both Eq. (4.31) and Eq. (4.32) include history data, analogous to that of an integrator, such that any steady state error is forced to zero. The choice of γ decides the convergence curve. The synchronisation error should be such that $e_2 << 1$ so the higher γ, the slower the convergence at very low instances of synchronisation error, thus the smoother the steady state values for the adaptive parameters but the less reactive it is to fast transient or frequency dependent plant behaviour. Additionally, as the adaption only occurs at the set trigger conditions, $\varphi_{1,\ldots,4}$, the AFP algorithm is subject to a persistence of excitation criterion [101].

To show the effectiveness of the AFP algorithm, we perform a system identification test again for each transfer system (from § 4.3.5) but with an excitation sweep of $r_{1,2} = 3$ to 10Hz in 5s and then back to 3Hz in 5s. This is a typical challenge faced in real-time substructuring. The rapidly changing transfer system dynamics mean that the algorithm must constantly adapt to achieve optimal synchronisation. Figure 4.11 shows the subspace plots for a sine sweep excitation test. The level of synchronisation for each transfer system is shown for the case when there is no forward prediction (only the inner-loop proprietary linear control is active) in Figures 4.11 (a1) and (b1), and for the case where we use the AFP algorithm as an outer-loop strategy in Figures 4.11 (a2) and (b2). It is clear that the use of the AFP algorithm results in a significant improvement in the level of synchronisation. The constantly changing conditions means that the delay compensation scheme never reaches steady state values but instead must constantly monitor the synchronisation

error in order to maintain a high level of compensation. Although the excitation demands are equal and opposite and the actuators are of the same type, we can see that the transfer system dynamics vary as shown by the differing adaption characteristics in Figure 4.12 (a). This is a marked difference to the similarity of the amplitude error shown in Figure 4.12 (b). Again, this highlights the need for the control of the transfer systems to be decoupled such that their specific mechanical characteristics can be dealt with independently.

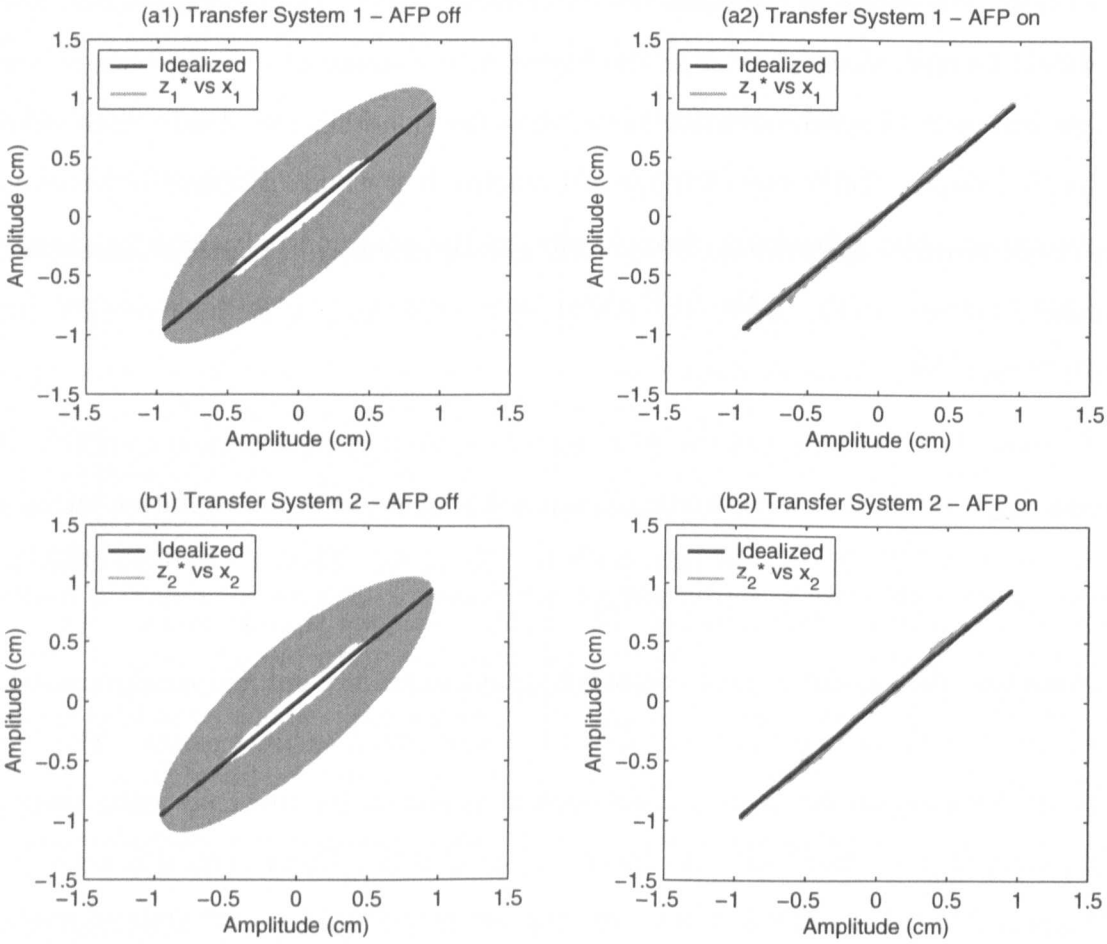


Figure 4.11: Synchronisation subplots for equal and opposite wall sweep excitation of $r_{1,2} = 3$ to 10Hz in 5s and then back to 3Hz in 5s. Controller parameters $N = 4$, $n = 10$, $\alpha_{1,2} = 100$, $\beta_{1,2} = 5$, $\gamma_{1,2} = 2$; Adaptive parameters shown in Figure 4.12.

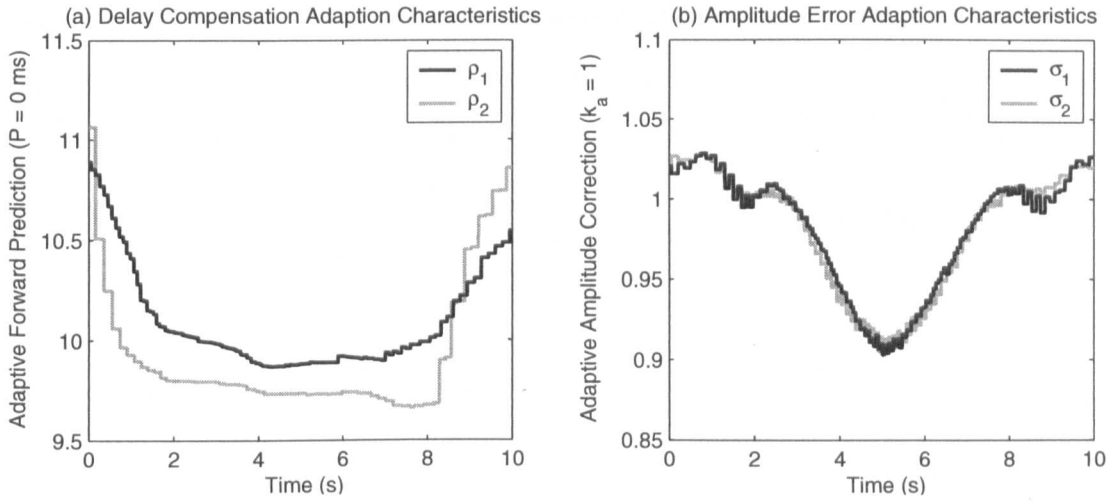


Figure 4.12: Adaptive parameter characteristics for Figure 4.11 (a2) and (b2).

4.3.7 Substructuring using the AFP algorithm

In order to demonstrate the effectiveness of the AFP algorithm in terms of an outer-loop strategy for substructuring we shall use the multi DOF case study of § 2.3.2 with various system parameters. In this way we can increase the demands on the controller by increasing the coupling effect of the substructure.

First, we look at the case of moderate coupling between the transfer systems. Moderately stiff springs are selected for the substructure of $k_{31,32} = 4750$ Nm^{-1} connected to a mass of $m_3 = 2.2$ kg. A system identification is performed to find damping constants of $c_{31,32} = 6$ Nsm^{-1} for the substructure. The same parameter values are used for the numerical models to make the system symmetrical. The stability of this system has been studied in § 3.2.5 and shown to have a critical delay value of $\tau_c = 2.5$ms. Figure 4.13 shows the steady state results for both transfer systems of an experimental substructuring test using the AFP algorithm for a test at 2Hz. Firstly, it can be seen that the substructuring algorithm has remained stable due to the high degree of synchronisation between the numerical models and their respective transfer systems, shown by Figure 4.13 (a2) for Transfer System 1 and Figure 4.13 (b2) for Transfer System 2. In both cases, virtually all the

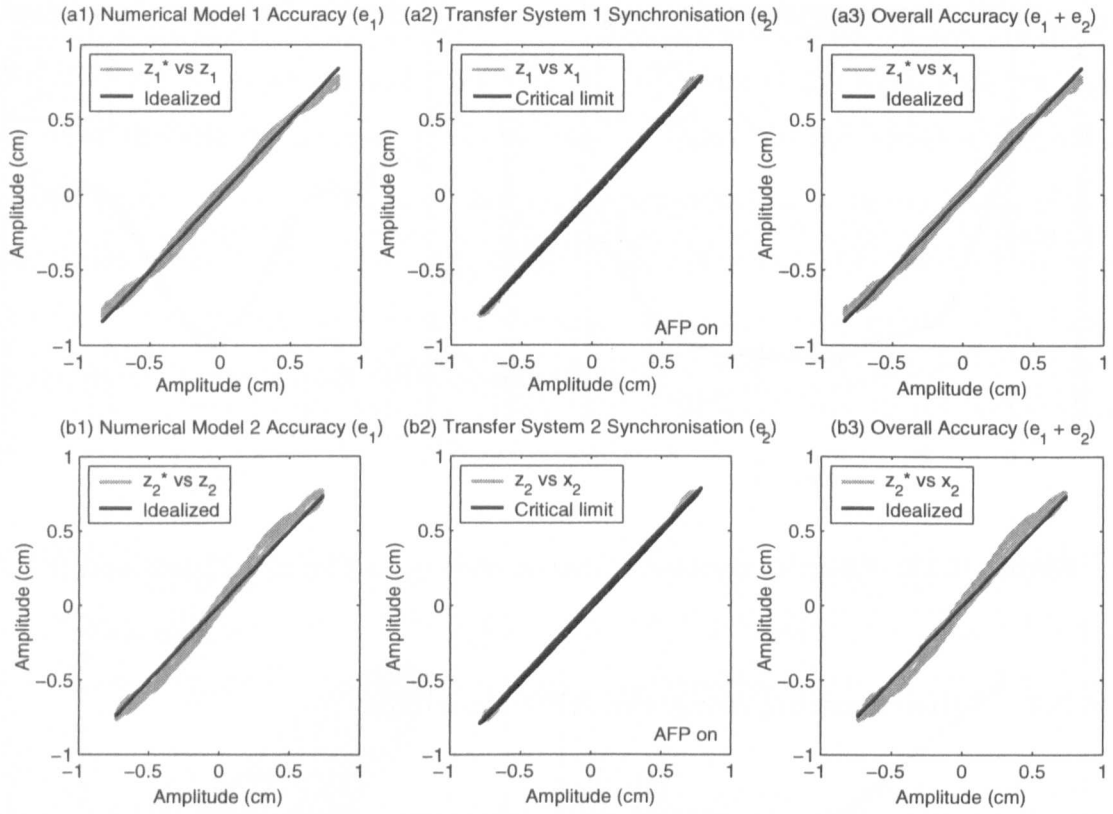


Figure 4.13: Comparative synchronisation subplots of a real substructuring test for wall excitation of $r_1 = 2Hz$ and $r_2 = 2Hz$ using the AFP algorithm; $\bar{\rho}_1 = 10.1$, $\bar{\rho}_2 = 9.6$, $\bar{\sigma}_1 = 0.99$, $\bar{\sigma}_2 = 0.98$, $\alpha_{1,2} = 10$, $\beta_{1,2} = 5$, $\gamma_{1,2} = 2$, $N = 4$, $n = 16$; System parameters: $m_{1,2,3} = 2.2$ kg, $k_{1,2,31,32} = 4750$ Nm^{-1} and $c_{1,2,31,32} = 6$ Nsm^{-1}; Critical delay: $\tau_c \approx 2.5$ms.

delay has been removed except for a slight increase in synchronisation error caused by the static friction of the actuator dynamics at the point of direction change, the *deadzones*. This synchronisation error, e_2, is called the *local* (control) error of the substructuring algorithm. The resulting numerical models can then be compared to their respective emulated dynamics as can be seen in Figures 4.13 (a1) and (b1). It can be seen that the synchronisation error, e_2, has had a direct influence on the accuracy of each numerical model — even though the substructuring algorithm remains stable there is still a *numerical model* error, e_1. This is highlighted by the overall substructuring error, called the *global* error, in Figures 4.13 (a3) and

(b3), which is a linear addition of the two sources of error, e_1 and e_2. However, as the coupling between the transfer systems is moderate in this case, below 50% of capacity, the global error is small thus giving an accurate substructuring result.

However, it is typical in real-time substructuring that the actuators are operated right up to their maximum performance envelope. It is close to and beyond this envelope that the cross-coupling and other nonlinear effects become significant. Therefore, we can repeat the previous test with different system parameters and at a higher velocity such that the coupling is significantly increased. The new

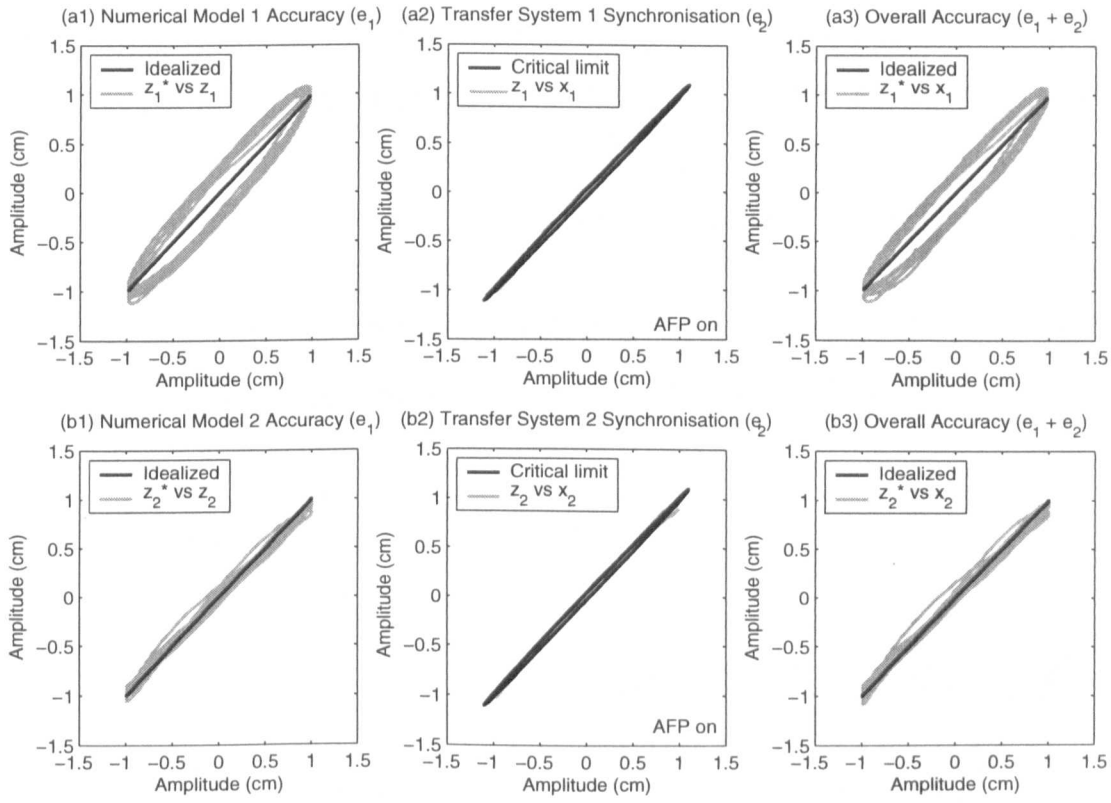


Figure 4.14: Comparative synchronisation subplots for a real substructuring test for wall excitation of $r_{1,2} = 8Hz$ (excitation magnitudes are not equal) using the AFP algorithm; $\bar{\rho}_1 = 9.67$, $\bar{\rho}_2 = 9.53$, $\bar{\sigma}_1 = 0.96$, $\bar{\sigma}_2 = 0.95$, $\alpha_{1,2} = 75$, $\beta_{1,2} = 5$, $\gamma_{1,2} = 2$, $N = 4$, $n = 10$; System parameters: $m_{1,2,3} = 2.2$ kg, $k_{1,2,31,32} = 9000$ Nm^{-1} and $c_{1,2,31,32} = 15$ Nsm^{-1}; Critical delay: $\tau_c \approx 3.3$ms.

system parameters are $m_{1,2,3} = 2.2$ kg, $k_{1,2,31,32} = 9000$ Nm^{-1} and $c_{1,2,31,32} = 15$ Nsm^{-1}. Performing a stability analysis for the current system parameters using DDE-BIFTOOL returns a critical delay of $\tau_c = 3.33$ms. Figure 4.14 shows the steady state results at an excitation frequency of 8Hz. Again, it can be seen that the substructuring algorithm has remained stable due to the high degree of synchronisation between the numerical models and their respective transfer systems, shown by Figure 4.14(a2) for Transfer System 1 and Figure 4.14(b2) for Transfer System 2. In both cases, the AFP algorithm has removed virtually all of the transfer system delay. However, there is a loss of accuracy in the resulting numerical models compared to their respective emulated dynamics, shown in Figures 4.14 (a1) and (b1), producing a large global substructuring error despite the small local control error. The reason for the extent of this substructuring error in this case is the magnitude of cross coupling between the transfer systems. Figure 4.15 shows the manufacture's specification for the actuator capacity envelope for the transfer systems used in these experiments. The experimental data shown is for Transfer System 1 (although a similar profile would be observed for Transfer System 2) for the test shown in Figure 4.14(a). It can be seen that the actuators are operated

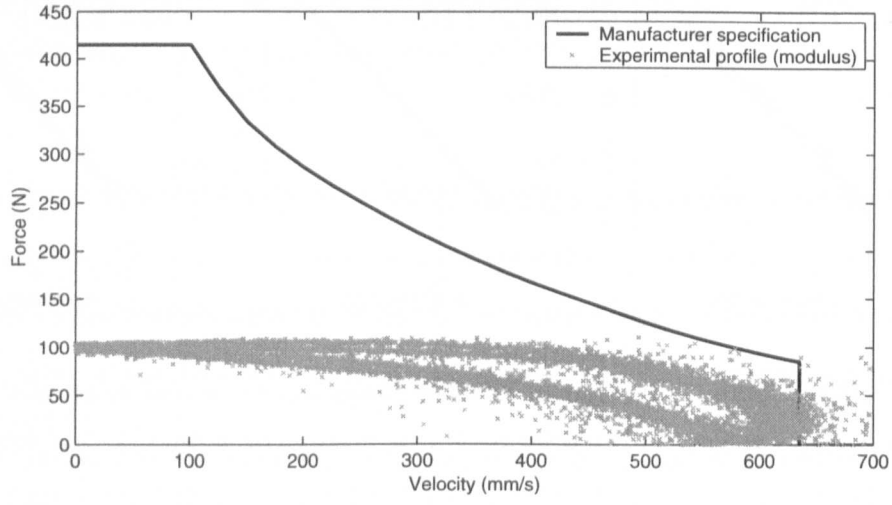


Figure 4.15: Actuator capacity envelope for Transfer System 1. Experimental data shown for the test of Figure 4.13(a) (10s test data).

right up to their maximum performance envelope which introduces significant errors into the substructuring algorithm resulting in the reduced global accuracy. However despite this, a good local synchronisation is maintained by the AFP algorithm.

However, we note that the numerical model error e_1 often cannot be calculated when performing a complex substructuring test as the emulated dynamics would not be known explicitly. Figure 4.16 shows the result of the global error in the dynamics of the substructure x_3 compared to that of the emulated system z_3^* for the example presented in Figure 4.14. The combination of the local control error and the numerical model error due to the cross coupling of the transfer systems has resulted in a non-linear relationship of the substructure dynamics compared to that of the emulated system. This is an important concept for measuring the accuracy of a substructuring test as the only error that we have a direct influence on is the level of synchronisation achieved, e_2. The magnitude of the numerical model error, e_1 and thus the resulting global error is dependent on the capacity of the transfer systems used to perform the test. Working well within the actuator's performance envelope will result in limited coupling between the transfer systems and therefore a correspondingly lower global error, Figure 4.13. For a complex substructured system this means that the explicit measure of accuracy, e_2 (the local control error),

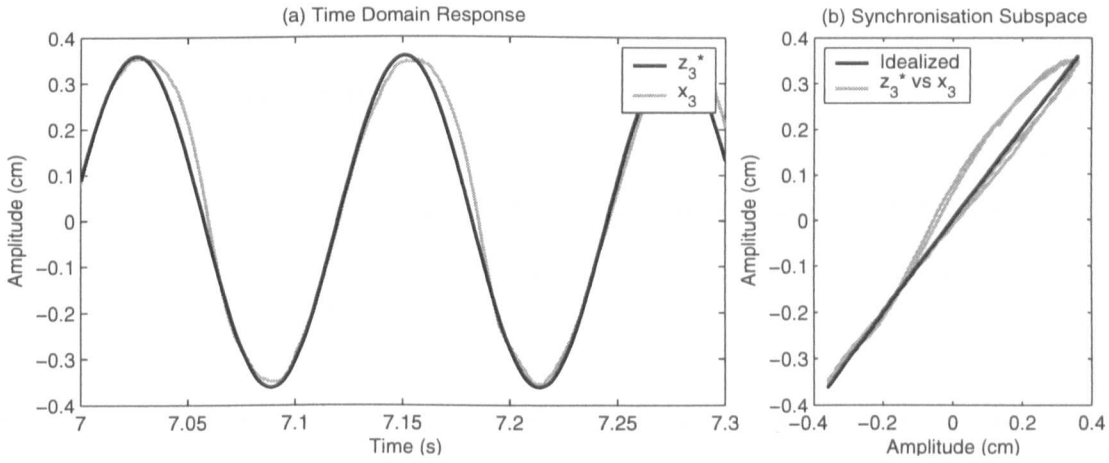


Figure 4.16: Substructure global accuracy for the test shown in Figure 4.14.

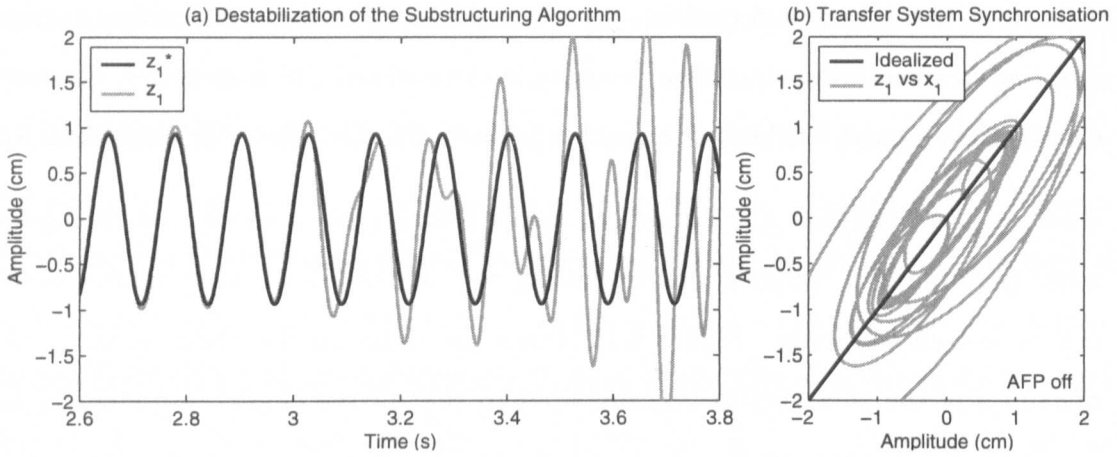


Figure 4.17: Destabilization of the substructuring algorithm for the test shown in Figure 4.14 using the inner-loop controller in isolation.

needs to be assessed in combination with the experimental transfer system profile (compared to its actual performance envelope) to gain a more complete measure of the accuracy of a substructuring test.

Figure 4.17 shows the case where the AFP algorithm is not used, such that the delay experienced due to a standard linear controller (the inner-loop P controller) is not removed. The substructuring test is started using the *emulated* force vector to ensure stability and then switched over to the *actual* force vector after approximately 3 seconds of run time. We observe the instability of the substructuring algorithm immediately building in Figure 4.17 (a). It is important to note that it is not the controller which is becoming unstable as can be seen from Figure 4.17 (b), which shows a consistent level of transfer system synchronisation throughout the entirety of the test. This example reconfirms that the removal of the transfer system delay is essential in ensuring a stable substructuring algorithm.

4.4 Lag compensation via a plant transfer function inversion

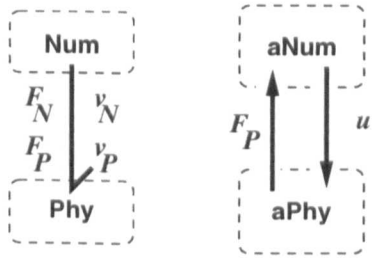
Lag compensation can be achieved by using an inverted model of the transfer system dynamics and has been proposed by Gawthrop et al. [76] and Sivaselvan et al.

[68]. This type of proposed outer-loop controller compensates for the unwanted dynamics by applying an inverse transfer function estimation of the transfer system to the numerical model in a similar vein to that of the delay compensation schemes. However, this method has the advantage that compensation of transfer-system dynamics is provided for all frequency ranges. Additionally, this approach can, in principle, be used in the case when the transfer system dynamics are nonlinear.

Additionally, model reference adaptive control has also been suggested as an outer-loop strategy by Wagg and Stoten [67], Neild et al. [77] and Lim et al. [78] which demonstrated how lag compensation can be achieved via this approach.

4.4.1 The virtual junction approach

Following the introduction to bond graphs in § 2.4.2, we can readily apply the technique to substructuring. Given the bond graph of a dynamic system, the set of components forming the physical substructure is chosen and all bonds external to this substructure marked, in general there will be $N \geq 1$ such bonds and the remaining components will form the numerical substructure. Thus each of the two substructures has N ports connected by the N marked bonds. In the case of mechanical systems, each port will correspond to a force-velocity pair; in general this can be any effort-flow pair.



(a) (b)

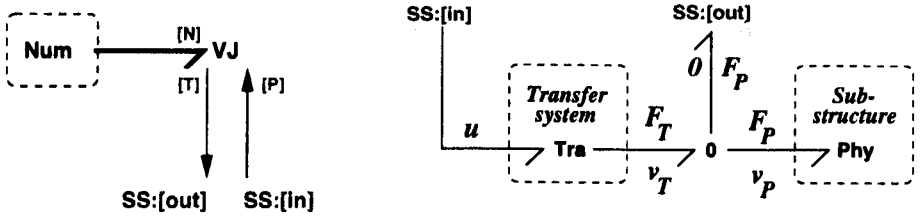
Figure 4.18: Substructuring: (a) Model; and (b) Augmented model

This decomposition is depicted in Figure 4.18(a) where **Num** and **Phy** are the numerical and physical substructures respectively. F_N and v_N are the force/velocity pair associated with the numerical substructure and F_P and v_P are the force/velocity pair associated with the physical substructure. The connecting bond implements the two *interface equations*:

$$\begin{cases} F_P &= F_N, \\ v_P &= v_N. \end{cases} \tag{4.33}$$

The energy bond of Figure 4.18(a) will, in general, be a *vector* bond corresponding to N scalar bonds and thus both the numerical and physical substructures may themselves contain many subsystems. In this case, the quantities in Eq. (4.33) can be regarded as vectors containing N components. The single DOF substructuring case study of § 2.3.1 has $N = 1$, where as the multi DOF case study of § 2.3.2 has $N = 2$.

As previously stated, it is not possible to connect the two substructures of Figure 4.18(a) because it is not physically possible to directly apply the signal implied by the numerical substructure to the physical system. In bond graph terms, the two systems of Figure 4.18(a) *cannot* be connected via an energy bond; as indicated in Figure 4.18(b), *augmented* versions of the numerical (**aNum**) and physical (**aPhy**) substructure are connected using a pair of active bonds. The augmented models are expanded in Figure 4.19. For the purposes of the work in this thesis it is assumed that:



(a) Numerical (**aNum**) (b) Physical (**aPhy**)

Figure 4.19: Augmented substructuring model

Assumption 2 *The causality is such that the* physical *substructure in Figure 4.18(a) imposes a* force *(in general effort) on the* numerical *substructure.*

Assumption 3 *The quantity imposed by the* physical *substructure in Figure 4.18(a) can be directly measured.*

Assumption 4 *The quantity imposed by the* numerical *substructure of Figure 4.18(a) cannot be directly imposed on the* physical *substructure but rather via an N-input* u transfer system. *In particular, the input* u *can only be imposed via a* transfer system *labelled* **Tra** *in Figure 4.19(b).*

Assumption 2 is not essential but simplifies the development; Assumption 3 is required for this work but can be negated by the use of observers; Assumption 4 is the main issue addressed here.

The fact that the substructured system of Figure 4.18(a) cannot be directly implemented but rather must be approximated by Figure 4.18(b) means that Eq. (4.33) no longer holds and must be replaced by:

$$\begin{cases} F_P - F_N &= \tilde{F}, \\ v_P - v_N &= \tilde{v}, \end{cases} \tag{4.34}$$

where $\tilde{F}$ is the *force synchronisation error* and $\tilde{v}$ is the *velocity synchronisation error.* The synchronisation problem is to reduce the two synchronisation errors to acceptable values, where *exact synchronisation* corresponds to

$$\begin{cases} \tilde{F} &= 0, \\ \tilde{v} &= 0. \end{cases} \tag{4.35}$$

In order to achieve synchronisation over the full range of experimental frequencies, the *virtual junction* approach to control system design can be applied. This technique was introduced by Gawthrop et al. [102] and experimentally verified by

Gawthrop [103] in the context of control system design. This technique was modified by Gawthrop et al. [76] such that it can now be applied to the substructuring problem; this section gives an overview of the approach.

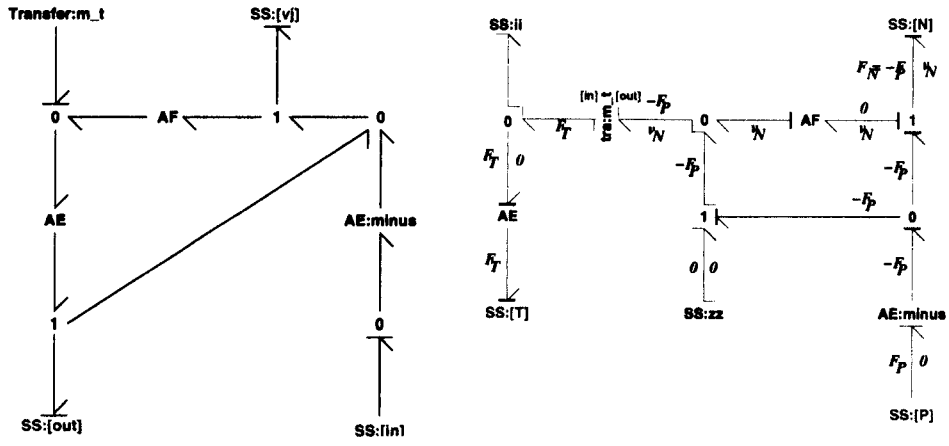
The *virtual junction* **VJ** component appearing in Figure 4.19(a) has three ports labelled:

[P] carrying the measured signal F from the *physical* system but imposing 0 signal onto the physical system;

[T] carrying the control signal u to the input of the *transfer* system but not carrying any measurement and

[N] the port to which the numerical system is attached.

The bond graph for the virtual junction is shown in both collocated and non-collocated form in Figure 4.20 (refer to Gawthrop et al. [102] for discussion on collocation). The purpose of the virtual junction is to make the input-output properties of the systems of Figures 4.18(a) and 4.18(b) identical. For example,



(a) Collocated (b) Non-collocated

Figure 4.20: Bond graph of the *virtual junction* (**VJ**).

this can be done if the virtual junction implements the equations:

$$\begin{pmatrix} F_N \\ F_T \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 1 & T^{-1} \end{pmatrix} \begin{pmatrix} F_P \\ v_N \end{pmatrix} \tag{4.36}$$

where $v_T = TF_T$. Given the structure of Figure 4.18, there are two restrictions on the class of systems for which a virtual junction can be successfully implemented:

Assumption 5 *The transfer system is stable and has stable zero dynamics.*

Assumption 6 *Defining σ_N as the length of the shortest causal path (SCP) between F_N and v_N and σ_T as the length of the SCP between F_T and v_T, then:*

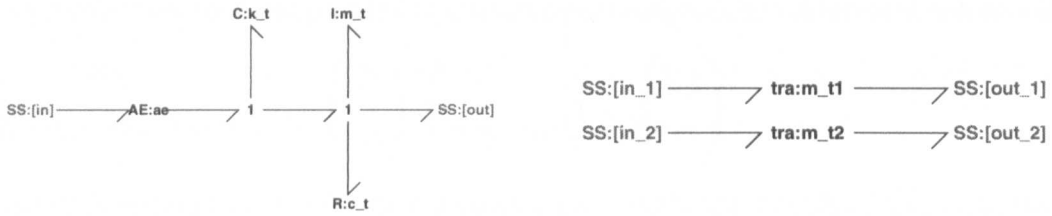
$$\sigma_N \geq \sigma_T \tag{4.37}$$

Assumption 5 ensures internal stability and Assumption 6 ensures that the combined numerical substructure and virtual junction is proper and thus has a state-space realisation. It is clear from this that an accurate model of the transfer system is required, the following section has more discussion on this point.

4.4.2 Transfer system identification

The virtual junction approach requires an accurate model of the transfer system to operate effectively, as can be seen from Eq. (4.36) where T is a transfer function describing the transfer system dynamics. The equipment used for experimentation is discussed in detail in § 2.3.3. In bond graph terms it is simplest to model the scalar transfer system **tra** as a mass-spring-damper unit, shown in Figure 4.21(a). Thus, the transfer function used in the virtual junction will be of the form $T = \frac{s}{m_t s^2 + c_t s + k_t}$. For a multi DOF substructuring system, multiple instances of **tra** are combined in to form a vector transfer system **Tra**. Figure 4.21(b) shows such a transfer system for the two-port system of the multi DOF case study (§ 2.3.2).

Unfortunately, AC servos are non-linear [104] and difficult to characterise from first principles as is the ball screw mechanism. Analysis of experimental measurements



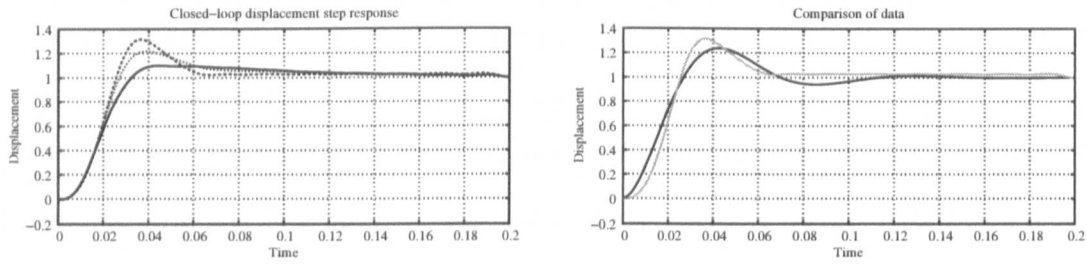
(a) (b)

Figure 4.21: Transfer system bond graph representation: (a) **tra** component parts; and (b) vector transfer system (**Tra**) for multi DOF case study.

of step responses showed that the dynamical response of the servo motor/ball-screw was indeed dependent on the form of the input signal. Most actuators used in this context are indeed non-linear and this must therefore be an important consideration in transfer system design. In particular, it is well-known that the use of feedback reduces uncertainty and nonlinearity. Thus, a variable-gain proportional digital controller (similar to that which will be used as the *inner*-loop controller in the substructuring algorithm) was implemented with gain k_{pi}, sample interval $\Delta t = 1$ms, demand x_{di} and described by

$$u_i = k_{pi}(x_{di} - x_i) \tag{4.38}$$

where u_i is the input to the transfer system and $i = [1, 2]$ depending on the actuator in question. The *closed-loop* transfer system was observed to have a more linear



(a) (b)

Figure 4.22: Identified transfer system (TS_1) step responses: (a) x_{t1} for $k_{p1} = 0.6, 0.8, 1.0$; and (b) x_{t1} (grey) and $\hat{x}_{t1}$ for $k_{p1} = 1.0$

response than the open-loop transfer system.

For transfer system 1 (TS_1), sequences of input (x_{d1}) and output (x_1) data were measured for three values of k_{p1}. Using the "frequency-sampling filter" method of Wang and Cluett [105], a step response (relating x_{d1} and x_1) was identified for each of the three values of gain and plotted in Figure 4.22(a). Although the bond graph of Figure 4.21(a) does not correspond in detail to the actual transfer system, it is *physically plausible* in the sense of Gawthrop [106] and so its parameters can be estimated using the *sensitivity* bond graph approach [107]. The mass m_{t1} is known, and so the two remaining parameters (spring stiffness k_{t1} and damping c_{t1} are identified and appear in Table 4.1. We must repeat this identification for the other transfer system (TS_2) as its frictional characteristics are not necessarily the same regardless of its mechanical similarity. In fact it can be seen from Table 4.1 that the dynamics from TS_2 are quite different.

The step response of this physically plausible model x_{t1} is compared with the data-based step response $\hat{x}_{t1}$ of Figure 4.22(a) for $k_{p1} = 1.0$ in Figure 4.22(b). The match is not perfect, but the identified model is a good representation of the linear transfer system dynamics. A similar result was found for the other transfer system.

Name	Value	Units
m_{t1}	2.2	kg
c_{t1}	150.8	Nsm^{-1}
k_{t1}	13618.0	Nm^{-1}
m_{t2}	2.2	kg
c_{t2}	200.8	Nsm^{-1}
k_{t2}	25902.0	Nm^{-1}

Table 4.1: Estimated transfer system parameters

4.4.3 Experimental validation

To show the effectiveness of the virtual junction as an *outer*-loop control strategy, we perform a simple system identification test at 5Hz as can be seen in Figure 4.23. The virtual junction is used in conjunction with an *inner*-loop P controller, with $k_{pi} = 1.0$ where $i = [1, 2]$ depending on the transfer system in question. To achieve synchronisation, the *actual* displacement of the transfer system, x_i, must be equal to the *desired* demand, z_i.

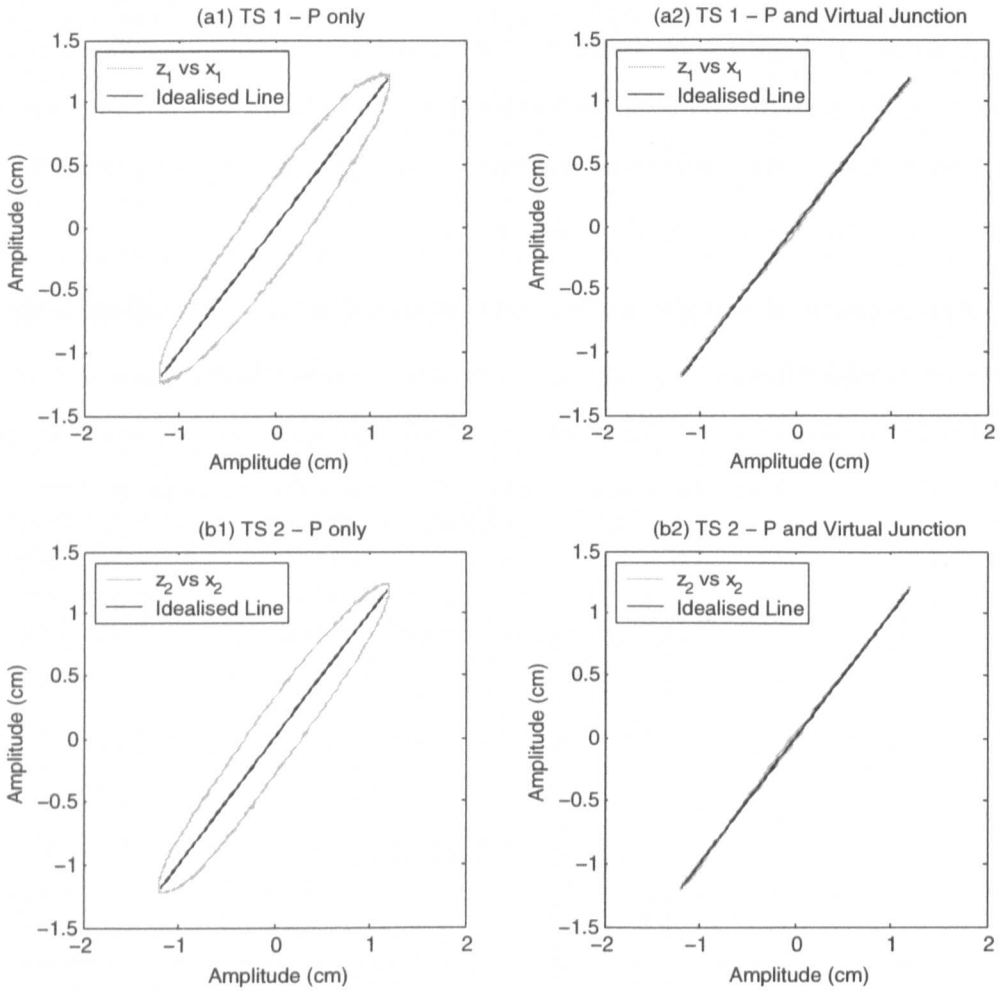


Figure 4.23: Synchronisation subplots for a system identification test for each transfer system at a frequency of excitation of $z_{1,2} = 5Hz$. Algorithm parameters shown in Table 4.1.

We again use *synchronisation subspace* plots (see § 2.5) to compare the effectiveness of the control algorithm when the virtual junction is utilised, Figure 4.23(a2) and (b2), to when the inner-loop controller is used in isolation, Figure 4.23(a1) and (b1). We can clearly see that the phase delay caused by the mechanical characteristics of each transfer system has been effectively removed by the inclusion of the virtual junction in both transfer systems. Comparing the shape of the subspace plots Figure 4.23(a1) and (b1) we can see that although the transfer systems are the same type of actuator they have slightly differing mechanical properties due to differing frictional characteristics as predicted by the transfer system identification in § 4.4.2. This again highlights why we must use a separate model for each transfer system in its respective virtual junctions to account for these mechanical variations.

4.4.4 Substructuring using the virtual junction

The transfer system models found though the system identification process (as described in § 4.4.2) are fixed throughout the test procedure. This effectively makes the phase inversion part of the control algorithm a feed-forward process, which means that it cannot cope with transient or unmodelled nonlinear behaviour. To show how the virtual junction can be used in the context of substructuring, we perform an experimental substructuring test but using forces generated by a model of the *emulated* system, rather than the actual *measured* forces. This will ensure stability as the numerical models will exactly match the dynamics of the emulated system, such that $e_1 = 0$. This is analogous to pseudodynamic substructuring where an estimation of the force is used Donea et al. [44]. The same system parameters of the moderate coupling example of § 4.3.7 are used for the following tests. Thus, a very small critical limit of $\tau_c \approx 2.5$ is defined.

Figure 4.24 shows the results when the wall excitations are not equal and opposite, thus the transfer systems must now be controlled to a compound sinusoid. We can see from Figure 4.24(a2) and (b2) that again the inclusion of the virtual junction

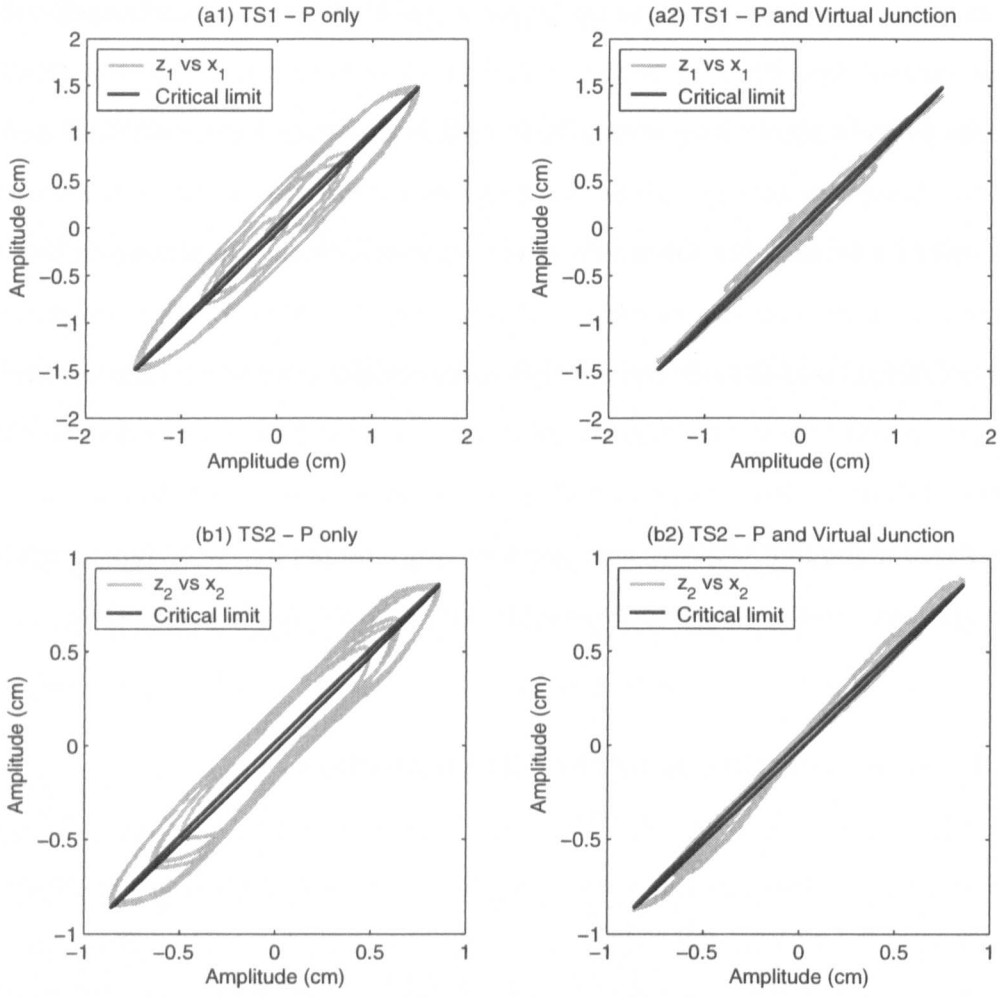


Figure 4.24: Wall excitation of $r_1 = 3Hz$ and $r_2 = 5Hz$; Virtual junction parameters given in Table 4.1; System parameters: $m_{1,2,3} = 2.2$ kg, $k_{1,2,31,32} = 4750$ Nm^{-1} and $c_{1,2,31,32} = 6$ Nsm^{-1}; Critical delay: $\tau_c \approx 2.5$ms.

has a beneficial effect on the synchronisation compared to when just the inner-loop P controller is used in isolation, Figure 4.24(a1) and (b1), but not to such an extent as in Figure 4.23. This is due to the transfer system models losing coherence at the low frequencies. This is the reason we must use the *emulated* force rather than the *actual* force as it is quite clear that the synchronisation of the transfer systems are both outside the critical limit. If the experimental force was used then this would lead to instability.

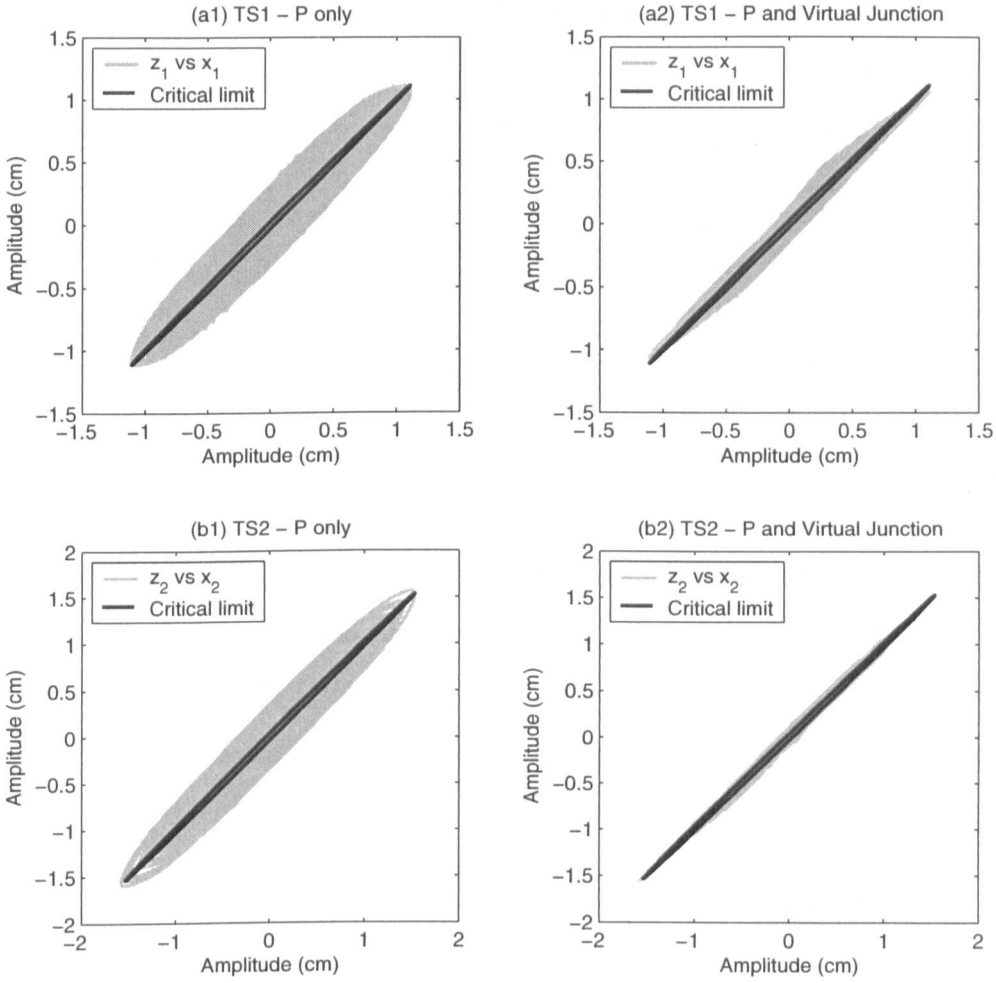


Figure 4.25: Wall sine sweep excitation of $r_1 = 1$ to 5Hz and $r_2 = 3$ to 4Hz in 60 seconds; Virtual junction parameters given in Table 4.1; System parameters: $m_{1,2,3} = 2.2$ kg, $k_{1,2,31,32} = 4750$ Nm^{-1} and $c_{1,2,31,32} = 6$ Nsm^{-1}; Critical delay: $\tau_c \approx 2.5$ms.

We can further see this when we introduce a sinusoidal sweep as the wall inputs. Figure 4.25 shows the case where we have a sweep from 1Hz to 5Hz for the left hand wall excitation, r_1, and a sweep from 3Hz to 4Hz for the right hand wall r_2. Although we again see a much higher level of synchronisation when the virtual junction is included in the numerical model, Figure 4.25(a2) and (b2), we cannot achieve the high level of coherence to achieve a stable algorithm if the experimental forces were used. This is again due to the nonlinear characteristics of the transfer systems not being fully captured by the model in the virtual junction for a very

small critical limit.

We can see from these results that the phase inversion achieved by the virtual junction has a significant effect on increasing the synchronisation of the transfer systems. However, the increase in synchronisation is not sufficient to maintain a stable substructuring algorithm if the actual measured forces were used in each test as can be observed from the synchronisation plots exceeding the superimposed limit of stability. If the damping was increased then it would become possible to achieve stable substructuring with the virtual junction, however, it is the typical situation in substructuring that the system has very low damping. The virtual junction's effectiveness could be significantly increased by replacing the transfer system models with an on-line system identification. This would close the control loop round the phase inversion stage of the virtual junction and make the it possible to achieve high levels of synchronisation for complex inputs, similar to that of the AFP algorithm.

4.5 Conclusion

In this chapter we have discussed two techniques for delay compensation and seen how they can be applied to the hybrid numerical-experimental technique of real-time dynamic substructuring. Both have been used to shown how the resulting delay errors from the inner-loop control of the transfer system(s) can be removed by using an online technique. In addition to stability, both techniques have benefits in achieving improved accuracy of the substructuring algorithm. This is in contrast to the immediate exponential growth observed when the test is carried out using a standard linear controller in isolation.

Both, lag and delay compensation techniques rely on creating a new reference signal forward in time from the original numerical model. They differ in the way the magnitude of this compensation is decided. The basic forward prediction algorithm and the virtual junction are similar in the fact that they are both fixed feed forward controllers, when used in conjunction with the inner-loop proprietary controller,

and thus both suffer from the an inability to compensate for nonlinear or transient behaviour. The AFP algorithm and the virtual junction are similar in the fact that they can both cope with frequency dependent plant behaviour. However, as the AFP algorithm does this in an adaptive manner (as it is effectively an error driven feedback controller when used as an outer-loop strategy) it does not rely on having an accurate model of the transfer system dynamics and thus has been shown to achieve higher levels of synchronisation in the substructuring tests. Additionally, it provides considerable advantages in terms of flexibility.

However, there are several ways in which the performance of the virtual junction could be improved. As highlighted in the experimental results, § 4.4.4, an accurate model of the transfer system is required. This can be directly improved by:

Transfer system modelling: A more sophisticated modelling process is required to capture the nonlinear characteristics of the transfer system. The bond graph approach can be used not only for modelling and control design but also for actuator sizing [108–110].

On-line System Identification: Replace the fixed models of the transfer system dynamics by an on-line system identification procedure. If this is evaluated as part of the numerical model stage then the transient behaviour and changing nonlinear plant conditions could be effectively controlled.

Although these could increase the performance of the virtual junction, the drawback is paid in terms of computational expense. In terms of achieving real-time control this can have a significant influence as it would restrict the sampling time choice. A major advantage of real-time substructuring over pseudo-dynamic is the typically very small sampling time (in comparison to the excitation frequencies) that allows explicit algorithms to be used (ensuring numerical stability of the integration scheme). Therefore, significantly increasing the sampling time is detrimental in real-time control. However, it is unnecessary to demand every requirement out of one

individual outer-loop algorithm. In fact, it can be beneficial to combine them into a formal strategy and use them as and when each is required, this is discussed in more detail in Chapter 5.

Additionally in this chapter, we have described a measure of the accuracy for a substructuring test without having to simulate the complete system. This is significant as it would be the typical situation in an industrial example of substructuring that the emulated dynamics are not known, and is in fact why the substructuring tests are being performed, the case in Chapter 6. The measure is a generic result which holds for all substructured systems, although the exact relationship between the *numerical model* error, e_1, and the *local* control error, e_2, will be system dependent.

Chapter 5

Robust substructuring

SUMMARY: This chapter extends the control strategy applied to real-time substructuring in order to achieve a more robust algorithm. Each technique reduces the uncertainty of the delay compensation scheme and increases the available margin to the critical limit of stability in compromise for reduced dynamical accuracy of the numerical model.

5.1 Introduction

From the previous chapters, it is clear that instability can still occur even in the presence of apparently quite small neglected dynamics; in other words, the outer-loop controller design is not necessarily robust. Robustness is an area that has been mainly overlooked so far in terms of substructuring. This could be due to the inherent reduction of the dynamical accuracy of the numerical model (to that of the emulated system) as a trade off for achieving a robust system. However, when the uncertainty (of the transfer system dynamics or the substructure response) is high or the margin to instability is very small it can be essential in achieving successful experimental substructure testing.

Broadly, the literature outlined so far addresses a) the selection of the inner-loop controller gains, b) a system identification of the resulting transfer system and c) the

design of an outer-loop controller to compensate for the transfer system dynamics. The adaptive nature of the outer-loop controllers proposed in [71, 73, 75] allow for the compensation of the transfer system dynamics despite uncertainty in the transfer system model. Although they incorporate some level of robustness due to this adaptation, they do not explicitly include a robustness compensator proposed as a separate constituent part.

Gawthrop et al. [74] proposed a four stage methodology to achieving a robust substructuring algorithm. Within this framework it is possible to combine a number of different types of controller and utilise them as and when they are required. This is the key reason why the linear results can be so readily applied to systems where the experimental substructure is nonlinear. The adaptive nature of these outer-loop controllers plus the robustness compensation allows the system to cope with a significant degree of nonlinear "disturbance". We note also that the analysis applies primarily to the design of the transfer system. In fact we wish to design a stable robust control strategy to eliminate (or at least mitigate) the effect of uncertainty and non-linearity from the transfer system — we want to make the transfer system dynamics linear. The only time we actually require a model of the (typically nonlinear) physical substructure is in order to apply the robustness compensation technique based on physical model emulation, as described in § 5.3.3.

In this chapter, the problem of robustness is considered for a generic substructuring system. In particular, robustness is a particular concern when the structure being tested is lightly damped or there is a high level of uncertainty in the transfer system control. Here we examine the robustness of a substructuring algorithm and describe a strategy for increasing the control robustness for lightly-damped systems. The effects of the robustness compensators are illustrated using the single DOF example, with both hybrid numerical-experimental results compared to pure numerical simulations, by intentionally increasing the uncertainty in the system.

5.2 Robust transfer system design methodology

Continuing the stability and robustness analysis of § 3.3 leads us onto a methodology for the *design* of the transfer system to achieve robust stability. The use of linear theory — and particularly the assumptions that the experimental substructure and transfer systems are approximated by linear transfer functions — could at first sight appear to be a serious limitation of this analysis. However, these results can be applied to — and in some cases can significantly improve results from — substructuring tests with nonlinear elements. Using this approach we are able to get a good comparison of results between three different types of robustness compensator, described in § 5.4.1 and § 5.4.2.

Using linear analysis, we propose a 4 stage controller design strategy for each transfer system (which in this work we assume to be an actuator) [74]:

1. Design an (or use the proprietary) inner-loop controller around the actuator to reduce uncertainty and nonlinearity in the resultant closed-loop transfer system response.

2. Use system identification to estimate a (closed-loop) transfer function of the actuator and inner-loop controller which we define as the *nominal* model. Use the same system identification results to estimate an *uncertainty* model for the transfer system.

3. Use the *nominal* model from Step 2 to design an outer-loop transfer system *cancellation* controller.

4. Use the *uncertainty* model from Step 2 to design a robustness compensator.

Broadly, Steps 1–3 are addressed in the literature outlined in Chapter 2 and 4. We now redesign our system identification of § 2.3.4 to fit into this strategy:

Step 1: Proprietary control. The proprietary control for the transfer systems was addressed in § 2.3.4. In summary, a linear proportional (P) controller is

used for the inner-loop control with a gain of $k_p = 1$ to give a slightly under damped system ($\zeta = 0.7 - 0.8$). In this way we achieve a system which has a fast transient and dynamic response, good repeatability, reduces the effects of nonlinearities and lowers the demand on the outer loop controller.

Step 2: Transfer system identification. A closed loop system identification with this gain setting was then completed, the results are shown in Figure 2.11 (§ 2.3.4). We observe the slightly underdamped response in Figure 2.11(a) as expected and then the dynamic characteristics of the transfer system in Figure 2.11(b) over the experimental range of frequencies. From this it is possible to construct our *nominal* and *uncertainty* model for Steps 3 and 4. In addition we observe that the response of the transfer systems can be approximated to a delay, rather than a lag (frequency dependent delay), for the experimental range of the actuators (0-15Hz).

Step 3: Nominal model. Using the experimental data from Figure 2.11(b) we are able to estimate our best fit nominal model — a transfer function that can replicate the dynamic characteristics of the transfer system the closest. We again use the Matlab Output Error (*oe*) function to approximate a 2^{nd} order fit of experimental response compared to the sine sweep demand, as can be seen below for each transfer system:

$$G_{n1}(s) = \frac{-480.9s + 5.651e004}{s^2 + 321.8s + 5.54e004}, \tag{5.1}$$

$$G_{n2}(s) = \frac{-396.8s + 4.553e004}{s^2 + 294.9s + 4.469e004}. \tag{5.2}$$

Focusing on Transfer System 1 (TS1), as both models are qualitatively similar, Figure 5.1 shows a comparison of the time domain responses of the experimental data taken for TS1, x, compared to its nominal model $G_{n1}(s)$, x_m, under the same excitation conditions. Figure 5.1(b) shows a synchronisation subspace plot of the *actual* response of the actuator against the *modelled* response for which we see a high level of synchronisation both in terms of amplitude and delay magnitude.

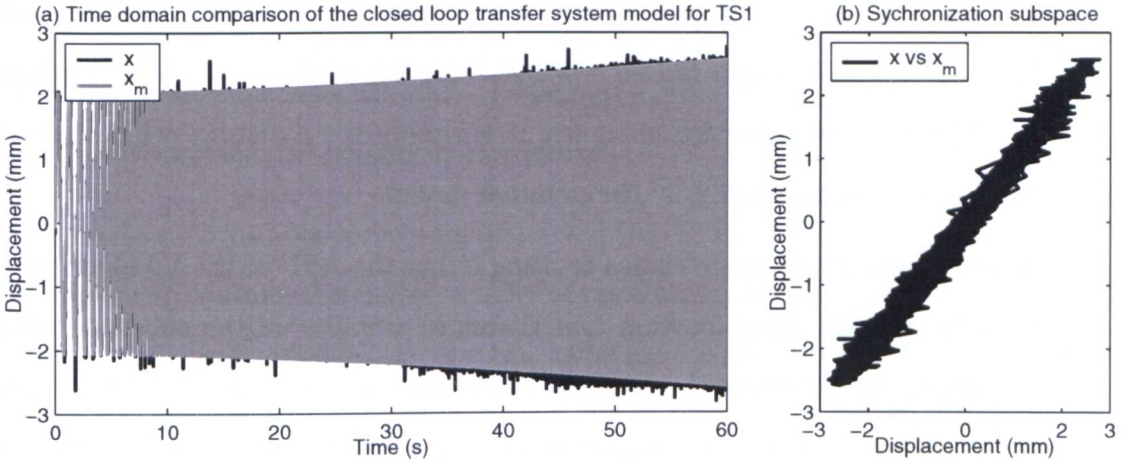


Figure 5.1: Comparison of the time domain responses of TS1 x compared to its nominal model x_m under the same excitation condition, r.

Figure 5.2 shows a close up view of Figure 5.1(a) at both high and low frequency of excitation in order to see how well the nominal model captures all the dynamics of the transfer system. Although there is a good match for both amplitude accuracy and delay magnitude for both conditions there is a significant difference in the magnitude of some of the nonlinear effects, expressly shown here by the large deadzone region at low frequency, Figure 5.2(a). We

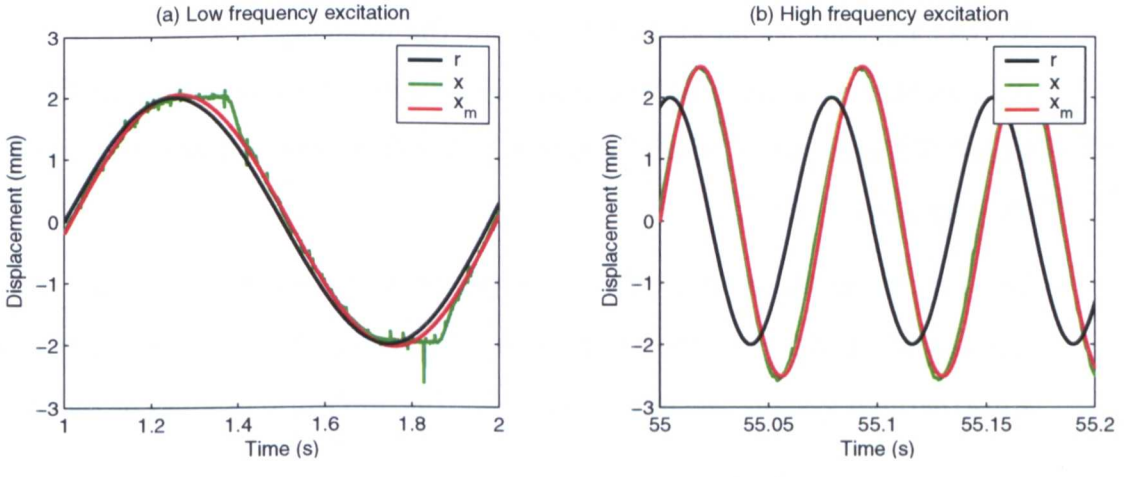


Figure 5.2: A comparison of the nonlinear errors experienced at low and high frequency excitation; experimental data: x, nominal model: x_m, excitation, r.

can therefore use this information to help create the uncertainty model — in this case the *uncertainty* will be repeatable for a given frequency and therefore should not be mistaken for *unknown*, it is simply the dynamics which cannot be modelled satisfactorily by the nominal model.

We now have an alternate choice to using either the AFP or the VJ algorithms to achieve delay compensation and therefore stability of the substructuring algorithm. By inverting the nominal model (as long as it can be made proper by combining it with the numerical model or built from scratch using the individual state signals) we, in effect, achieve a gross feed-forward cancellation controller which removes all the modelled dynamics from the transfer system such that:

$$z(t) = \frac{z}{G_n} G_a \approx z(t), \tag{5.3}$$

where, G_a is the actual dynamics of the transfer system and G_n is its nominal model. This is in fact directly analogous to the virtual junction approach (§ 4.4.1) to delay compensation but using transfer function representation rather than a bond graph one.

Step 4: Uncertainty model. Depending on the critical limit of stability, the remaining dynamics may or may not send the substructured system unstable. We cannot compute an explicit model for the unmodelled dynamics as they are expressly nonlinear, but we must be aware of when they are significant. Figure 5.2(a) shows us that this is true at low frequency at low amplitude for these transfer systems

We describe in the next section three potential robustness techniques to be used as an additional outer-loop control strategy. When analysing a new substructured system, the most suitable for the application should be selected.

Depending on the purpose, the equipment being used and the type of test being performed the controllers of Step 3 & 4 can be mixed and matched as necessary. For example, this highlights the versatility of the AFP algorithm — it can either be used

as the cancellation controller of Step 3 and/or the robustness compensator of Step 4 at the same time, refer to § 5.5. In fact, the outer-loop compensators discussed by [71, 73] can also be utilized in this format.

When selecting the appropriate algorithm for Step 3, the *cancellation* controller, it is important to consider the characteristics of the transfer system closely. In the case of the experimental equipment used so far, inverting the second order transfer function of Eq. (5.1) will require all three physical states of the transfer system (displacement, velocity and acceleration) in order to achieve the appropriate cancellation. While we can take advantage of the numerical model to output these states, the higher the order required the more experiential noise (which is also inherently fed back from the substructure) is amplified. Additionally, the transfer system is not straightforward as it describes the complex mechanical internal workings of the actuator, which is one reason the virtual junction approach could not achieve a sufficient level of compensation in § 4.4.4. This model inversion process is really only applicable to simple first order systems, such as hydraulic actuators, where although the capacity is typically much higher the effects of nonlinear characteristics are far smaller.

5.3 Robustness compensation techniques

We present here three approaches for the robustness compensator, Step 4 of the robust transfer system design methodology. The trade off for achieving a robust system is a reduced level of accuracy in the numerical model, the magnitude of which is discussed in § 5.3.4 and experimentally shown in § 5.4.1 and § 5.4.2. The following are the proposed strategies:

1. Phase-advance compensation: **α-robustness**

2. Damping-ratio compensation: **ζ-robustness**

3. Physical model emulation: **γ-robustness**

All of these approaches have a single parameter which provides a trade-off between performance and robustness and are discussed in the following sections § 5.3.1-5.3.3. Methods 2 and 3 are believed to be new and have been developed specifically with substructuring in mind [74]; Method 1 is a standard control system technique [99] but applied here for the first time to substructuring. Method 3 is related to the Youla parametrisation of all stabilising controllers, Goodwin et al. [99].

5.3.1 Phase-advance compensation: α-robustness

The stability of the example given in § 3.2 and § 3.3 implies that the lack of robustness is due to the neglected phase lag associated with $\Lambda(j\omega)$ at $w = \omega_c$. One way to improve robustness is to deliberately introduce phase advance, at the critical frequency, to $\Lambda(j\omega)$ by interposing a phase advance transfer function between **num** and **tra**. The simplest such transfer function is [99]:

$$C(s) = \frac{\alpha s + \omega_c}{\frac{1}{\alpha} s + \omega_c} \tag{5.4}$$

where the parameter $\alpha \geq 1$. Clearly $\alpha = 1$ corresponds to a unit transfer function which has no effect; the maximum phase advance occurs at about $\omega = \omega_c$. The maximum phase advance rises to about $\frac{\pi}{4}\text{rad} = 45^o$ when $\alpha \approx 10$. Typically, for this application, $1 < \alpha < 2$.

This method has the advantage of requiring only knowledge of ω_c, but has the disadvantage of distorting the closed loop system.

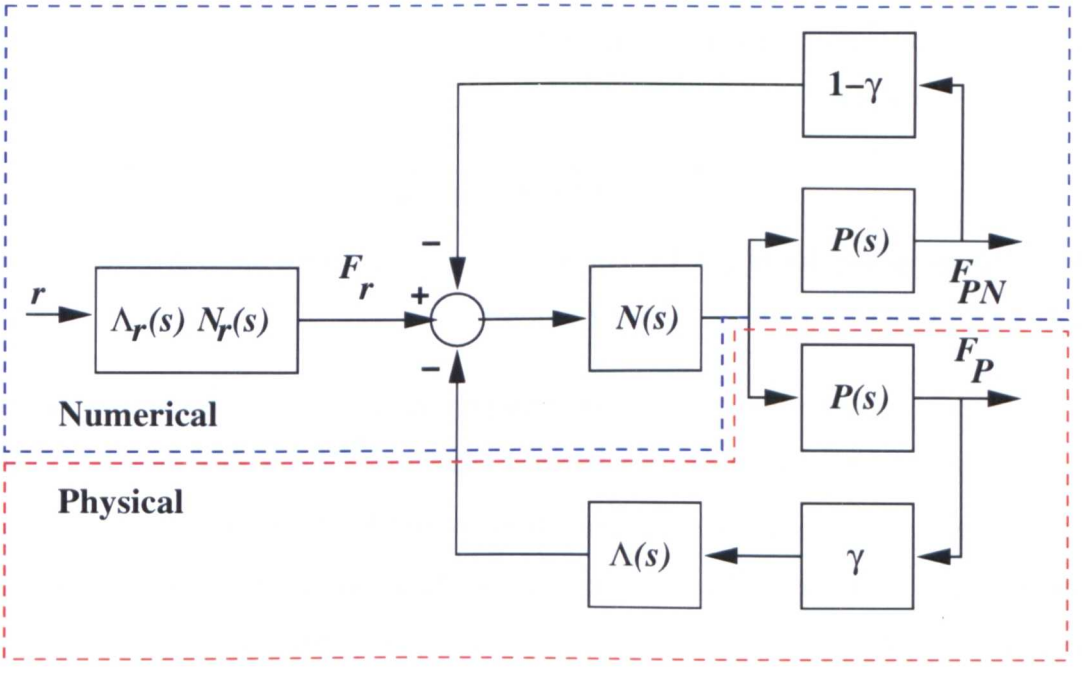
5.3.2 Damping-ratio compensation: ζ-robustness

From Figure 3.10(a) (§ 3.3.3), it is clear that it is the magnitude of the resonant peak of the closed-loop transfer function $|D_0(j\omega_c)|$ that restricts the maximum allowed value of uncertainty $|\Lambda(j\omega_c)|$. As $|D_0(j\omega_c)|$ decreases with increasing damping, a simple way of trading robustness for stability is to increase the damping coefficients of the numerical model above their correct values.

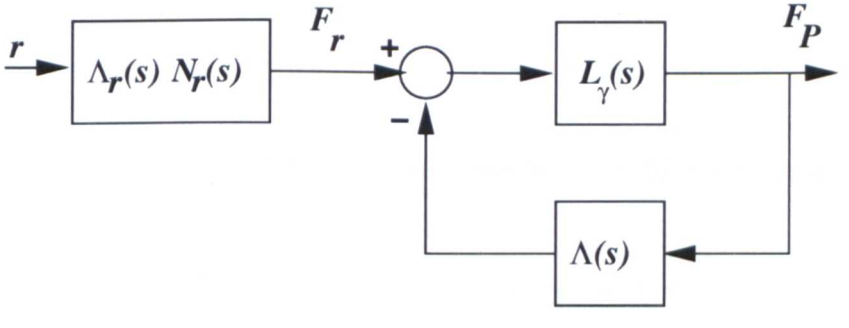
This method has the advantage of requiring no knowledge about the system properties but has the disadvantage of again distorting the closed loop system.

5.3.3 Physical model emulation: γ-robustness

In addition to the numerical simulation of the **num** subsystem, this approach also simulates the **phy** subsystem. As indicated in block diagram form in Figure 5.3(a), the output of **num** is fed into both the simulation of **phy** and the physical subsystem



(a) Block diagram



(b) Robustness block diagram

Figure 5.3: γ-robustness.

phy. In comparison to Figure 3.7 (§ 3.3.1), there are *two* feedback loops: γ times the output of the physical system and $(1-\gamma)$ times the output of the simulated physical system is fed back to the input of the numerical subsystem. At the two extremes, $\gamma = 0$ gives a purely numerical algorithm (and $\Lambda(s)$ is not part of the feedback), whereas $\gamma = 1$ gives the full hybrid numerical-physical algorithm that is subject to stability criteria discussed in Chapter 3. When $0 \leq \gamma \leq 1$, there is a smooth transition between the two extremes. However, for each value of γ, the nominal closed loop system dynamics are the same given an accurate model of the substructure.

The block diagram of Figure 5.3(a) can be rewritten in the simplified form of Figure 5.3(b) where

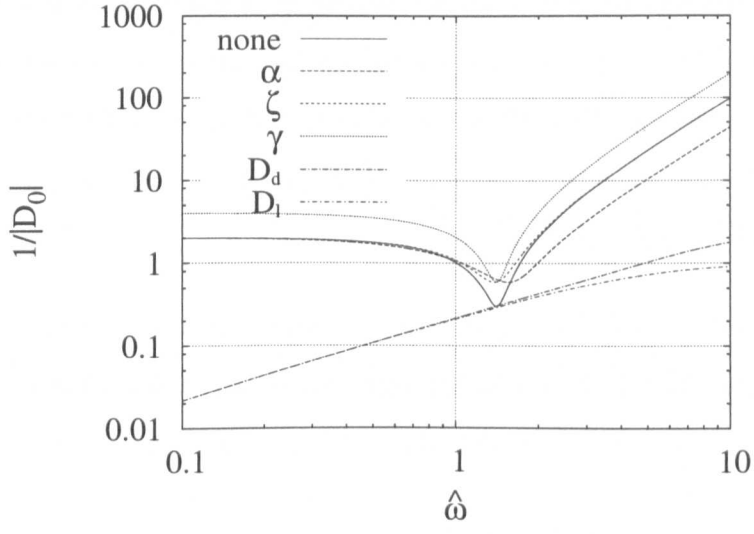
$$L_\gamma(s) = \frac{\gamma L_0(s)}{1 + (1-\gamma)L_0(s)}. \tag{5.5}$$

The robustness results of § 3.3.1 can then be applied to Figure 5.3(b) in a similar way as to that discussed for Figure 3.7 (§ 3.3.1).

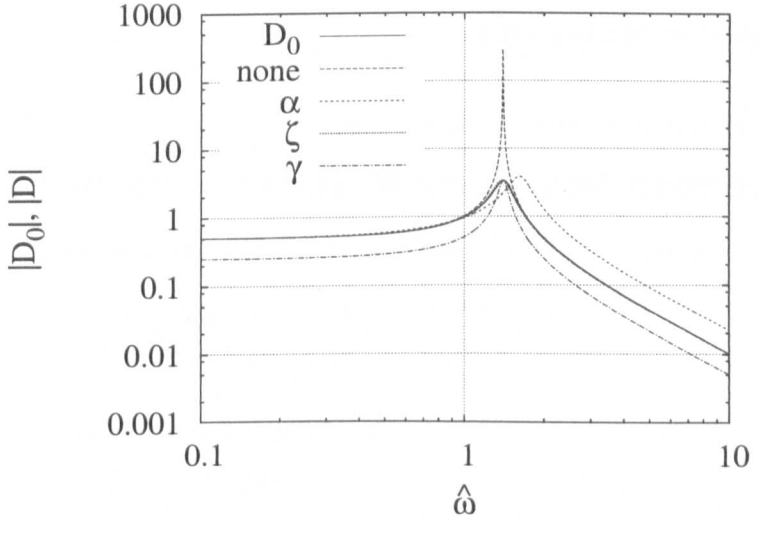
This method has the disadvantage of requiring an accurate model of the physical system transfer function $P(s)$ but has the advantage of *not* distorting the closed loop system. For complex systems, as described in Chapter 6, only an approximate model of the substructure may be possible and thus distort the closed loop system. However, the system dynamics should be qualitatively similar such that the transition to $\gamma = 1$ can be made without exciting any unwanted/erroneous transient effects.

5.3.4 Comparison of the robustness compensators

Figure 5.4(a) is the same as Figure 3.10(a) (§ 3.3.3) except that the results of each of the three compensators are shown as well. In each case, the stability margin is significantly increased at the resonant frequency; in each case, $\hat{\tau} \approx 0.35$, about $\frac{0.35}{0.21} = 1.67$ larger than the uncompensated case.



(a) Robustness analysis



(b) Closed-loop response

Figure 5.4: Robustness Compensation. (a) The inverse magnitude of $D(j\hat{\omega})$ is plotted against $j\hat{\omega}$ on a logarithmic scale for the three robustness compensators with $\alpha = 1.5$, $\zeta_r = 2\zeta$ and $\gamma = 0.5$. For comparison two possible uncertainty transfer functions $e^{-j\hat{\omega}\hat{\tau}}$ and $\frac{1}{1+j\hat{\omega}\hat{\tau}}$ are plotted for $\hat{\tau} = 0.35$. (b) The lines marked α, ζ and γ give the corresponding closed-loop systems for each compensator, in the presence of $e^{-j\hat{\omega}\hat{\tau}}$. The case of no compensator, with (none) and without (D_0) delay, is given for comparison.

Figure 5.4(b) shows how the nominal model of the closed loop transfer system (including the time delay) is compensated by the three robustness compensators. Although, as expected, these three closed-loop responses differ from the nominal, D_0, they are better than the uncompensated case which displays a large resonant peak indicating near-instability.

The other observation from Figure 5.4(a) is that there are significant differences between $1/|D|$ for the three different compensation methods. This will be discussed in more detail in relation to the hybrid numerical-experimental results shown in the next section.

5.4 Induced uncertainty

The proposed robustness methods were evaluated using the single DOF example of § 2.3.1. The physical substructure had of stiffness $k_s = 2250\text{Nm}^{-1}$ connected to a numerical model with $\zeta = 0.1066$ damping. The experimental equipment has been extensively analysed, see § 5.2, and it is known that a good model for the neglected dynamics of the system under an inner-loop proportional (P) control with $k_p = 1$ is a pure delay $\Lambda(\hat{s}) \approx e^{-\hat{s}\hat{\tau}}$ where $\hat{\tau} \approx 0.29$. Using the parameters for this example and Eq. (3.41) from § 3.3.2 the critical value of the delay is found to be $\hat{\tau}_c \approx 0.2$. Therefore this system is *unstable* without some form of delay compensation because $\hat{\tau} > \hat{\tau}_c$, thus we employ the AFP algorithm of § 4.3.6 as Step 3 of the robust transfer system design. This can then be used to achieve a unit gain response of the transfer system such that $\hat{\tau}$ is in the range $0 \leq \hat{\tau} < \hat{\tau}_c$.

5.4.1 Delay uncertainty

To highlight the effects of the three different robustness compensators, we are able to select a $\hat{\tau}$ value as appropriate within the stable range of $0 \leq \hat{\tau} < \hat{\tau}_c$ using the AFP controller by altering the synchronisation origin. This is a simple way of varying

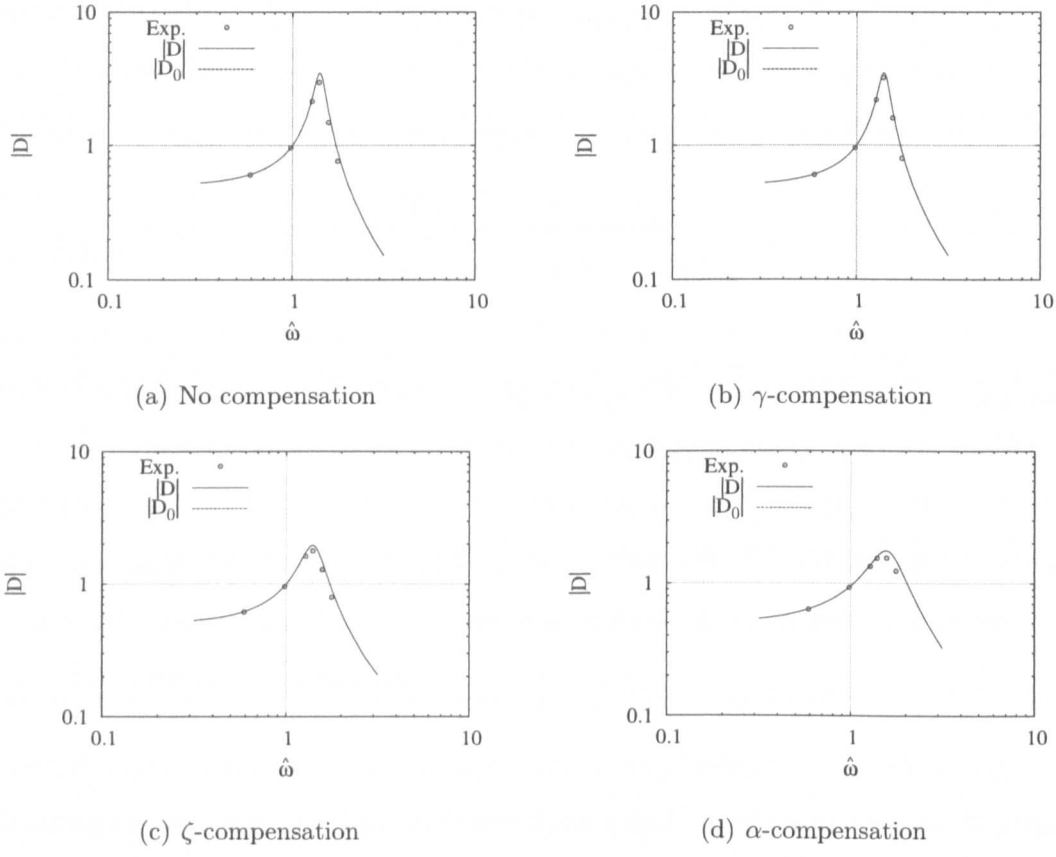
Figure	$\hat{\tau}$	$\frac{\hat{\tau}}{\hat{\tau}_c}$
5.5	0	0
5.6	0.1	0.5
5.7	0.19	0.95

Table 5.1: Experiment Summary

the degrees of uncertainty in the system, and gives an indication of the performance of each robustness compensator as the degree of uncertainty increases. Table 5.1 summaries the values of $\hat{\tau}$ used to generate the hybrid numerical-experimental results shown in Figures 5.5–5.7. For each $\hat{\tau}$ value, Table 5.1 shows the ratio $\hat{\tau}/\hat{\tau}_c$ to give an indication of how close the system is to the stability boundary at $\hat{\tau}/\hat{\tau}_c = 1$.

Each of Figures 5.5–5.7 shows hybrid numerical-experimental results for four cases: no compensation, γ-compensation, ζ-compensation and α-compensation. Each plot shows three sets of data. Circles correspond to hybrid numerical-experimental measurements of $|D(j\hat{\omega})|$ Eq. (3.35) at six frequencies: 3.0Hz, 5.0Hz, 6.5Hz, 7.1Hz, 8.0Hz and 9.0Hz. The solid and dashed lines are the theoretical values of $|D(j\hat{\omega})|$ (with $\Lambda(\hat{s}) \approx e^{-\hat{s}\hat{\tau}}$) Eq. (3.35) and $|D_0(j\hat{\omega})|$ (where $\Lambda(\hat{s}) = 1$) Eq. (3.37) from § 3.3.1 respectively.

Figure 5.5 shows the case where $\hat{\tau} = 0$ (i.e. the delay compensation method is removing the full 9.4ms of delay in the system). In this case $|D(j\hat{\omega})|$ and $|D_0(j\hat{\omega})|$ are indistinguishable and the uncertainty is very low such that $\Lambda \approx 1$ and $|D| \approx |D_0|$. With no compensation (Figure 5.5 (a)) there is good agreement between the hybrid numerical-experimental results and $|D_0(j\hat{\omega})|$, indicating (as expected) that for this hybrid test setup the delay compensation method of [75] provides a significant degree of robustness without an additional compensator. Figure 5.5 (b) shows the γ-compensation case where there is no distortion of $|D_0(j\hat{\omega})|$, and close agreement with the hybrid results. Figures 5.5 (c) and (d) show the ζ-compensation and α-compensation respectively. In each case the robustness is improved — indicated by an increased stability margin — but $|D_0(j\hat{\omega})|$ is distorted. Agreement with the



(a) No compensation

(b) γ-compensation

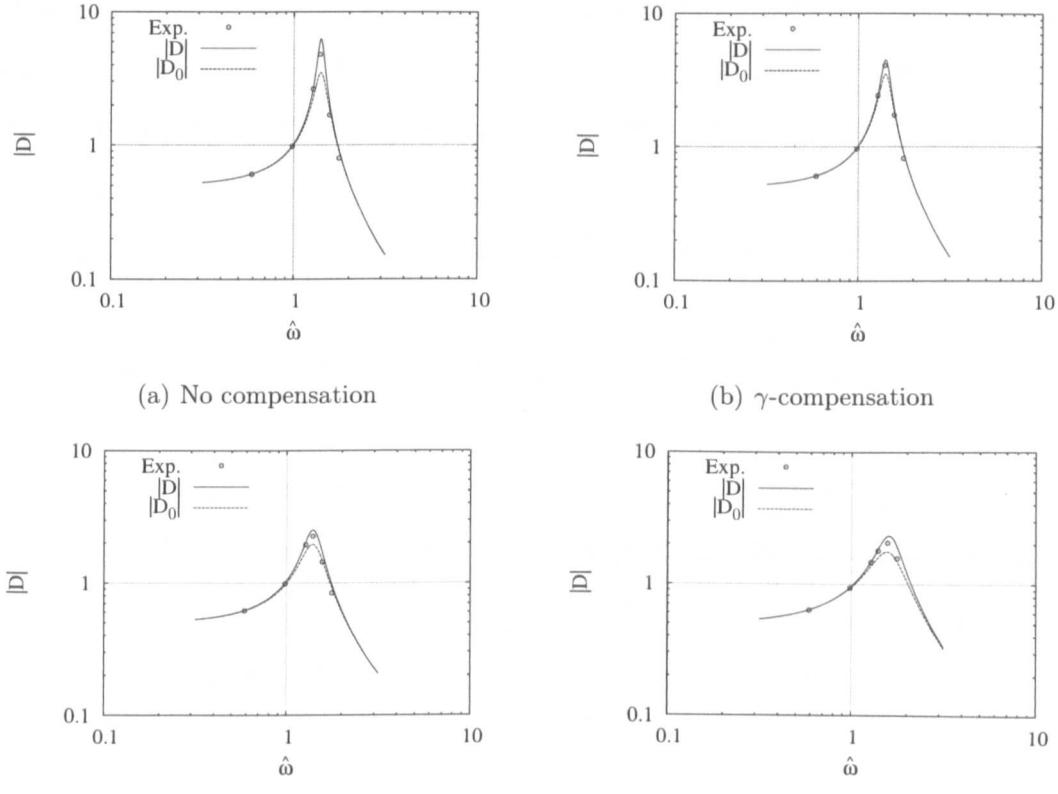
(c) ζ-compensation

(d) α-compensation

Figure 5.5: Experimental Results: $\hat{\tau} = 0.00$. In this case the uncertainty is very low so $\Lambda \approx 1$ and $|D| \approx |D_0|$ in all four cases. γ-compensation does not distort $|D|$ but ζ and α-compensation do. The experimental fit is good in each case.

hybrid test results is good although the α-compensation looses some correlation near resonance.

Figure 5.6 shows the case where $\hat{\tau} = 0.1$. This case corresponds to the situation when the delay compensation method is not fully compensating for the delay error. This can be seen in Figures 5.6 (a)-(d) as the discrepancy between $|D|$ and $|D_0|$ close to resonance. Now without any robustness compensation, the resonance peak $|D(j\hat{\omega})|$ becomes significantly exaggerated near the resonant frequency compared to the nominal case $|D_0(j\hat{\omega})|$. In Figures 5.6 (b)-(d) the three robustness compensators results are shown. The γ-compensation results give a significant improvement in reducing $|D(j\hat{\omega})|$ to $|D_0(j\hat{\omega})|$. The ζ-compensation and α-compensation also achieve



(a) No compensation

(b) γ-compensation

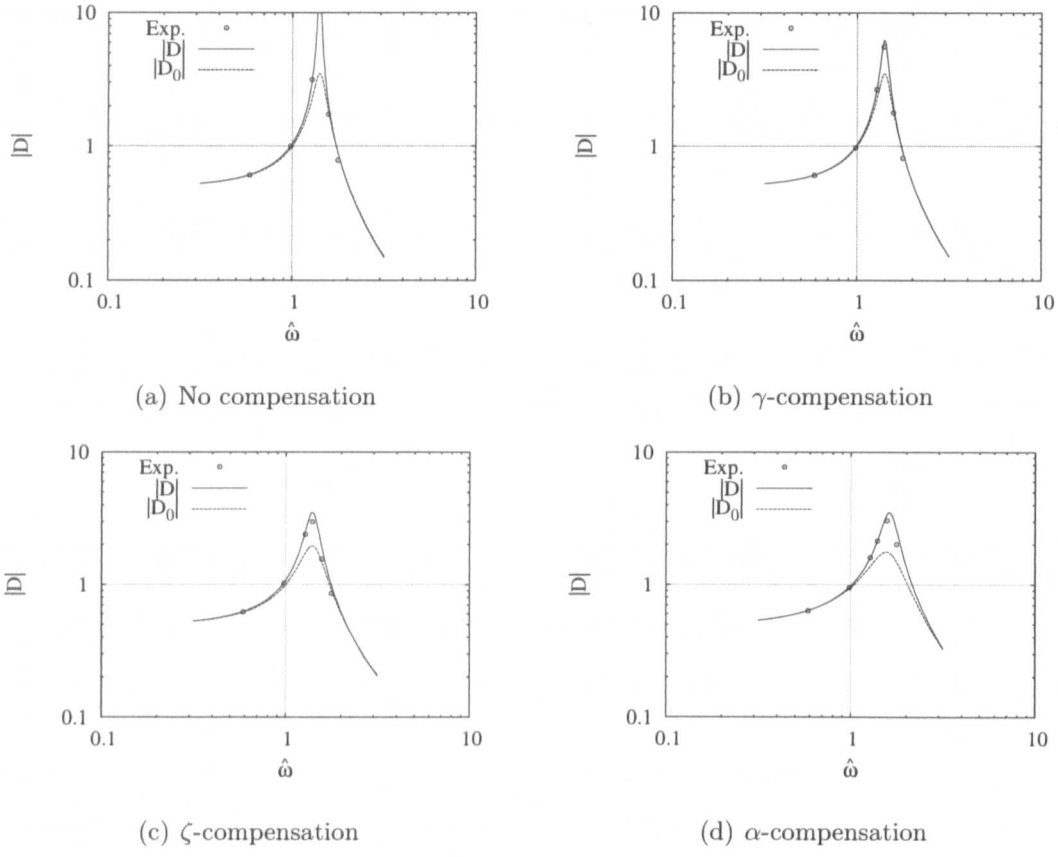
(c) ζ-compensation

(d) α-compensation

Figure 5.6: Experimental Results: $\hat{\tau} = 0.10$. There is a small amount of uncertainty due to the neglected delay so $\Lambda \neq 1$ and $|D| \neq |D_0|$ in each case. No compensation leads to an exaggerated resonant peak which is reduced by each of the three compensators.The experimental fit is good in each case.

the same effect, but with significant distortions in the $|D_0(j\hat{\omega})|$ transfer function. In all three compensation cases the hybrid results match well with $|D(j\hat{\omega})|$.

Figure 5.7 shows the case where $\hat{\tau} = 0.19$. This case corresponds to the situation when the delay compensation method is stabilising the system, but leaving a significant delay error — corresponding to a higher degree of uncertainty in the system. This can be seen clearly in Figure 5.7 (a) where now the discrepancy between $|D(j\hat{\omega})|$ and $|D_0(j\hat{\omega})|$ is even more pronounced close to resonance. The compensation methods shown in Figures 5.7 (b)–(d) all help to reduce this significantly. As with the previous example the hybrid results correlate well with $|D(j\hat{\omega})|$ across the frequency



(a) No compensation

(b) γ-compensation

(c) ζ-compensation

(d) α-compensation

Figure 5.7: Experimental Results: $\hat{\tau} = 0.19$. There is a large amount of uncertainty due to the neglected delay so $\Lambda \neq 1$ and $|D| \neq |D_0|$ in each case. No compensation leads to an almost unstable system with almost no damping and an excessive resonant peak which far from the nominal. Each of the three compensators stabilises the system giving a peak much closer to the nominal. The experimental fit is good in each case.

range considered.

5.4.2 Model uncertainty

In § 5.4.1 we explicitly know the level of uncertainty (in terms of the magnitude of transfer system delay remaining) as we know a good nominal model for the transfer system to be a pure delay of 9.4ms when the controller gain is set at $k_p = 1$. The consequence of reducing the accuracy of this nominal model, such

that a controller gain of $k_p = 0.5$ was used for example, would be to introduce an unknown level of uncertainty into the substructuring algorithm. As a result, an the unknown magnitude of delay will now remain in the system. The need for robustness compensation is therefore greater as the margin to instability is unknown. This is the typical situation when the dynamics of the transfer system are complex — we can never have 100% confidence in the accuracy of the nominal model so therefore, when the margin to instability is small the need for robustness increases due to the level of unknown/unmodelled dynamics. As our understanding of the system increases the level of robustness can be reduced until it has zero effect, the ideal situation. This can be done either during the initialisation of an experiential test (and then kept at a constant level) or more likely, changed online such that it is only utilized when it is needed as discussed in the next section.

5.5 An over compensation strategy using the AFP algorithm

A fundamental difficulty for substructuring is that it is only safe to start an experimental test from a region of stability, otherwise the unstable growth may make the test impossible or damage the substructure. This is of particular importance when the substructured system is lightly damped as the magnitude of unstable growth will be larger. Due to the versatility of the AFP algorithm we can formulate a strategy that combines both robustness and accuracy for such a situation. If we think of delay as adding negative damping in our system [61], then over compensating (predicting too far forward in time) will have the opposite effect of increasing the damping. This will give a similar kind of damping to that of the ζ -compensator (§ 5.3.2) but combining Steps 3 & 4 of the robust transfer system design methodology.

It is therefore a major concern for the performance of any delay compensation algorithm to find the interval of permissible magnitudes of compensation where the substructured system is stable. Pragmatically, when starting a substructuring

test for the case $\tau > \tau_c$ we can initiate the test using a numerical estimation of the force (γ-compensator, § 5.3.3) and switch over to the measured force when the control algorithm has achieved a high level of synchronisation. However, ideally the test should be initiated using the measured force itself to retain the true structural characteristics.

To highlight the merit of this approach, we use the very lightly damped single DOF system of § 3.4.1. This system only has a damping value of $\zeta = 0.0213$ which gives the system a critical delay of $\tau_c = 1.335$ms when $p = 1$ ($k = k_s = 2250$N/m). Figure 5.8 demonstrates the stability restrictions imposed on the AFP algorithm by the DDE system for the two pairs of algorithm parameters N and n for a variable ρ magnitude. To this end, we use DDE-BIFTOOL to find the eigenvalues of the DDE model explicitly including the delay compensation scheme to find the forward prediction stability regions. Figure 5.8(a) represents the stability of the AFP algorithm for a fitting polynomial $P_{N,n,\Delta t}$ of order $N = 2$ for $n = 10$ previous values of z, while Figure 5.8(b) corresponds to a polynomial of order $N = 4$ fitted to $n = 16$ previous values ($\Delta t = 1$ms). Both prediction schemes are compared to the exact prediction (grey line) of $F = -k_s z(t + \rho - \tau)$ within the interval from -20 ms to 45 ms. If the real part of the eigenvalue is positive then the system is unstable. The dashed lines highlight the parameter value where the forward prediction equals the actual delay in the system, $\rho = \tau = 9.4$ms. The polynomial forward prediction gives, in general, only a finite interval of stability for ρ. For low order schemes the interval of permissible ρ is large. Stability ranges from $\rho \approx \tau - \tau_c = 8.065$ms to $\rho_{max} \approx 41$ ms for $N = 2$, $n = 10$ (as in Figure 5.8(a1)). This defines the other critical limit, the forward critical limit τ_f as discussed in § 2.1 Table 2.1. Thus, for low N the AFP algorithm can start with an initial guess for P that is substantially larger than the delay τ. Increasing the order N of the fitting polynomial improves the accuracy of the prediction as can be seen from the improved frequency accuracy of the polynomial fit shown in Figure 5.8(b2) compared to Figure 5.8(a2). However, in general, this shrinks the range of forward prediction ρ that is permissible for

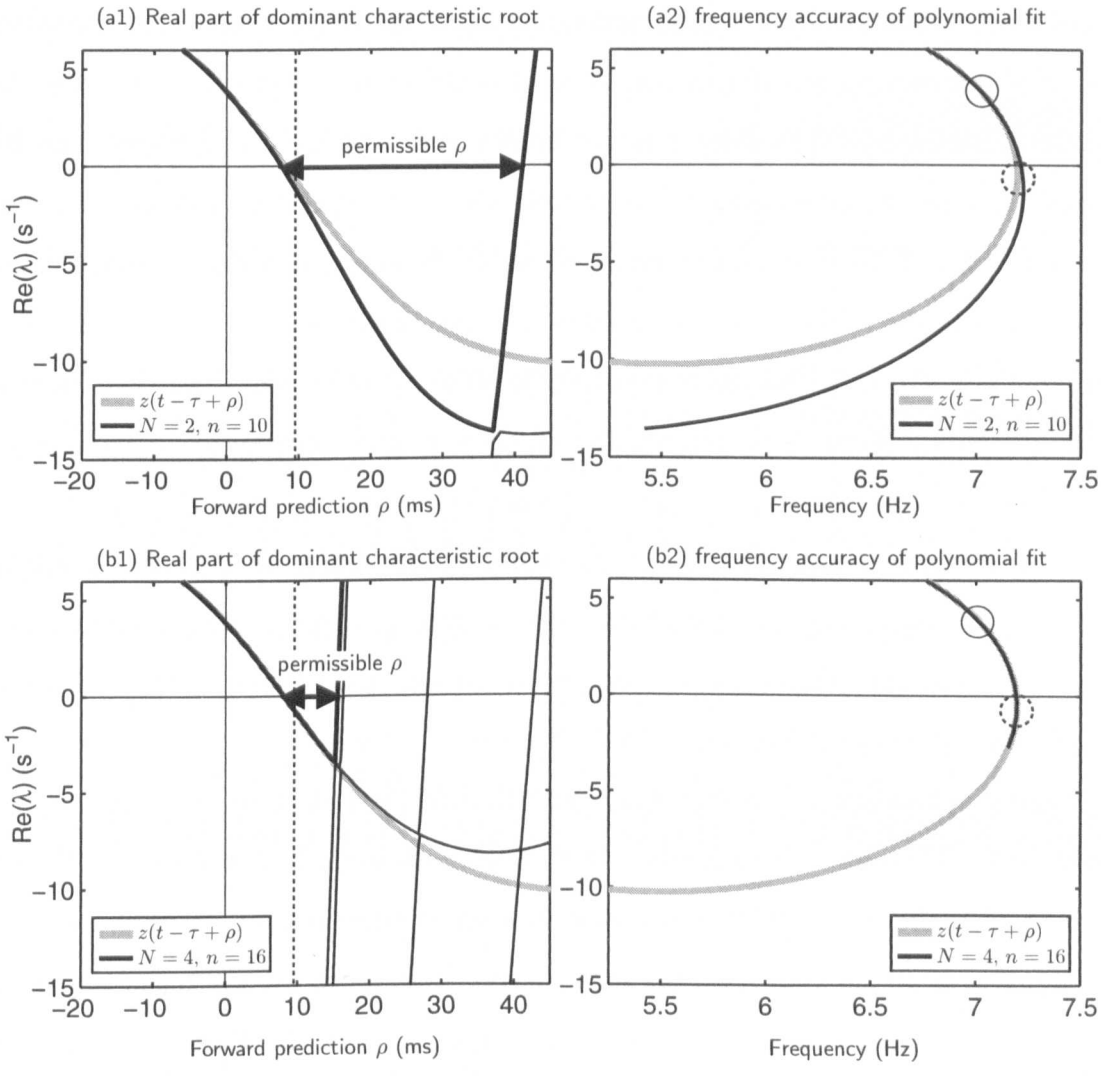


Figure 5.8: Eigenvalues for two delay compensation schemes compared to the exact value $z(t-\tau+\rho)$ (grey line); dominant eigenvalue is highlighted in bold. The dashed lines (circles on frequency plots) highlight where the forward prediction equals the actual delay of the transfer system, $\rho = \tau = 9.4$ms.

stability giving a new τ_f. Figure 5.8(b1) shows that the maximal permissible ρ is at $\rho_{\max} \approx 15$ ms for $N = 4$, $n = 16$. Near τ_f another eigenvalue of the system becomes dominant and unstable but is dependent on the delay compensation parameters and/or the system properties, rather than the system alone as for τ_c.

Obviously, it is preferable to start the test with the optimum level of compensation,

but when we do not have a good understanding of the substructure characteristics, or of the transfer system(s) we are using, it could be very difficult to estimate this value. Therefore, if the delay τ is not known and expected to be larger than the critical delay of the substructured system, the AFP algorithm should start with a low order N to give a large range of stable forward prediction ρ and to over-compensate the initial guess, as this will give the largest stable region as shown by Figure 5.8. Once the adaption algorithm is close to convergence, we then increase the prediction order N to improve the accuracy of the substructuring experiment. We note that the permissible order of N is also limited by the noise fed back from the load transducer as well as $\rho_{max} > \tau - \tau_c$. Additionally, if delay is equivalent to adding negative damping in our system, then over-compensating (predicting too far forward in time) will have the opposite effect by increasing the damping. If we control to a *shifted* synchronisation origin, such that we now take $\tau = -1$ms as having zero synchronisation error, for example, this will have the effect of over-damping the dynamic response of the numerical model (analogous to the ζ-compensator). Firstly, this makes the numerical model slower to react to sudden state changes, i.e., high frequency noise fed back from the substructure, and, secondly, will mean that there is greater margin before the critical delay limit is reached. This can be seen in Figure 5.9, where panels (a) shows the case of no delay compensation, (b) the case for under-compensated (zero initial compensation) and (c) the over-compensation method with shifted synchronisation origin of $\tau = -1$ms.

It is noteworthy that although under-compensated, the substructured system re-gained stability after approximately 2s in Figure 5.9(b). However, once stable the unwanted oscillations take a further 3s to die out. It is not always possible for stability to be regained, in this case the controller adapts faster than the unstable growth. The consequences of such unknown dynamics could significantly affect the structural properties of the substructure and possibly lead to failure. The over-compensation method is advantageous as the substructuring algorithm is always stable (under the maximum limit) as shown in Figure 5.9(c2), and therefore, has a correspondingly

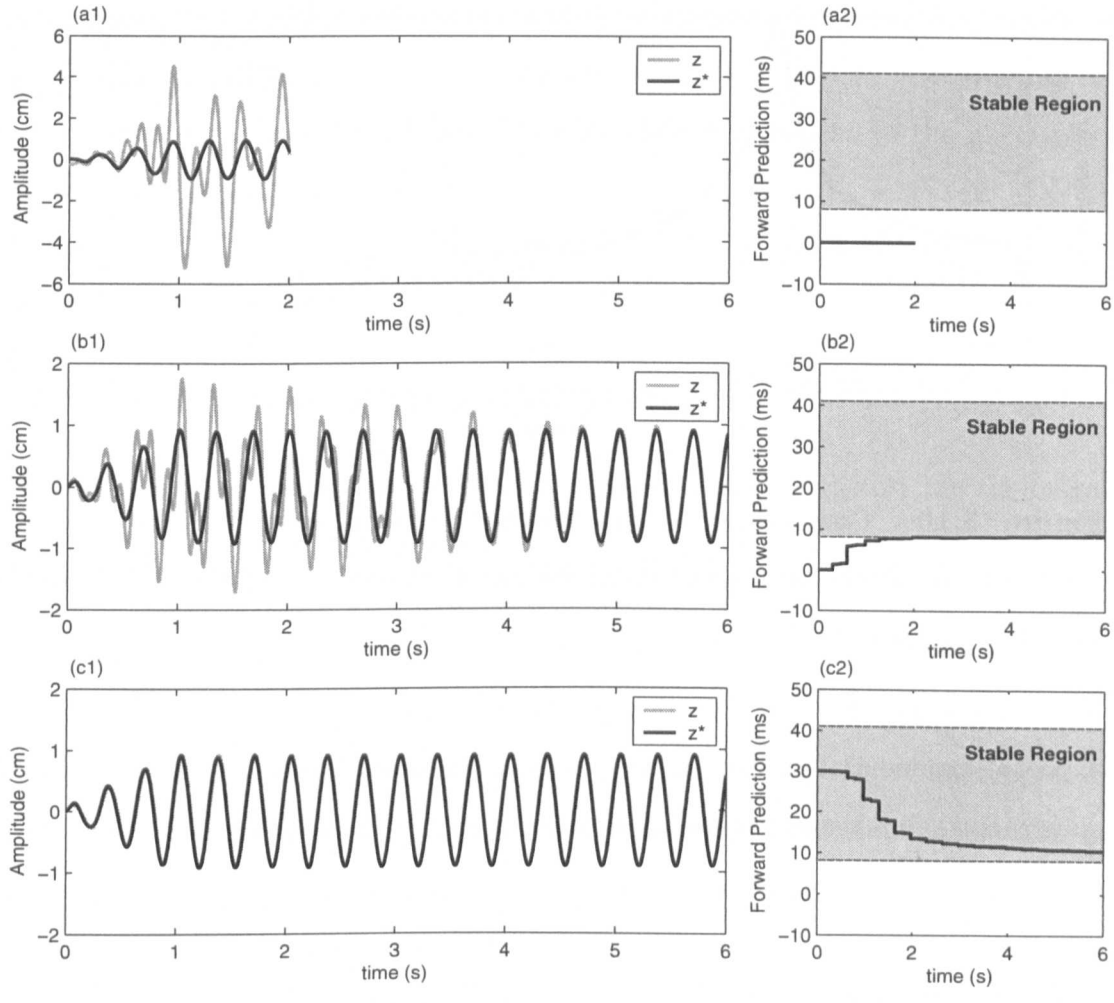


Figure 5.9: Experimental numerical model accuracy for differing control methodologies: (a) no delay compensation, (b) under-compensated (zero initial compensation), (c) over-compensation method (shifted synchronisation origin of $\tau = -1$ms). Controller parameters are $N = 2$, $n = 10$.

high numerical model accuracy throughout, Figure 5.9(c1). The frequency at which instability is observed is shown in Figure 3.13. There is little difference in the spectral frequency response between the emulated dynamics (calculated numerically) and the over-compensation method (measured experimentally) whereas there is an increase of approximately three orders of magnitude at a frequency of 7.1Hz when the substructured system is under-compensated ($\omega_I = 7.1$Hz from Eq. (3.22)). This is in contrast to the case of no delay compensation for which we see the

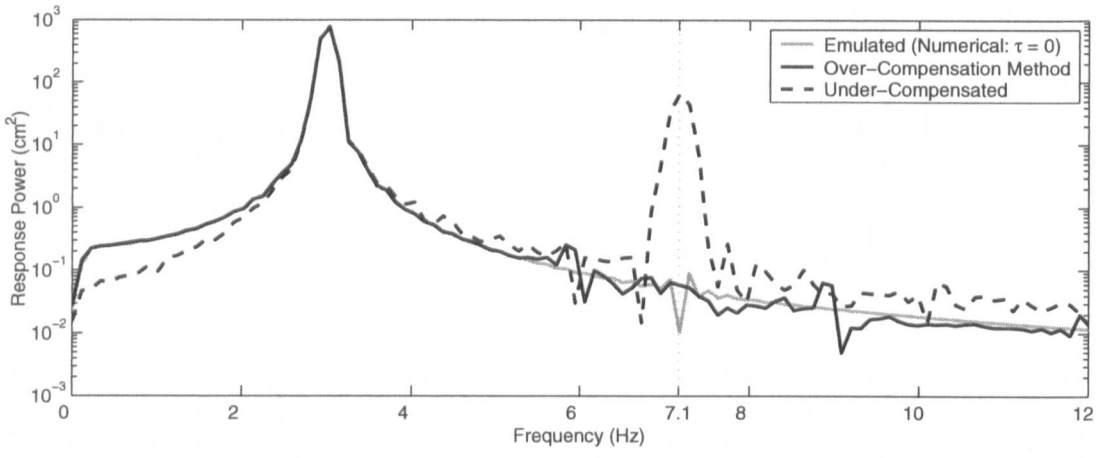


Figure 5.10: Comparison of the frequency spectrum for the experimental substructured responses (6s test data, see panels (b) and (c) of Figure 5.9) to the Emulated system.

expected exponential growth and necessity to terminate the test after 2s to prevent substructure damage in Figure 5.9(a1).

5.6 Conclusion

It has been shown that the hybrid simulation of lightly-damped dynamical systems using numerical-experimental real-time substructuring is sensitive to both transfer system delay and uncertainty. The four stage robust transfer system design methodology presented in here is designed to reduce both of these destabilizing effects. Three methods for reducing the effect of uncertainty are given: γ-compensation, ζ-compensation and α-compensation. We use both theoretical and experimental results to show that each is effective in increasing robustness to uncertainty. Each has the following strengths and weaknesses:

γ-compensation has the advantage of not modifying the overall system response but is based on having an accurate model of the physical system — this was available for the experiment considered here but would not generally be the

case. In less well known experimental equipment, details of $\Lambda(\hat{s})$ would be unknown and thus initial experiments would use $\gamma = 0$. γ could then be increased as more experimental results allowed reduction of the uncertainty encapsulated in $\Lambda(\hat{s})$.

ζ-compensation does not require a model of the physical system but does change the overall system characteristics; however, it has a clear physical meaning as a numerical model with an increased damping ratio.

α-compensation again does not require a model but does distort the the overall system characteristics significantly.

A pragmatic view of robustness is that the amount of compensation would be large for initial experiments, but would reduce as uncertainty was reduced. For example, using an advanced system identification technique as suggested by Gawthrop et al. [76] or when using an adaptive cancellation controller for Step 3 of the robust transfer system design. It should be noted that the lower the damping in the system the greater the destabilizing effect both the transfer system delay and uncertainty have on the substructuring algorithm.

Furthermore, we have proposed an additional strategy that can combine both the *cancellation* controller and the *robustness* controller by making use of further DDE analysis of the system using DDE-BIFTOOL. The proposed over-compensation method for substructuring has a number of distinct benefits. The margin to the critical limit of stability is extend and the effects of high frequency noise are reduced. Additionally, the test can be initiated from and entirely operated within a stable region of the substructuring algorithm. These factors become increasingly important as the damping of the structure is reduced or the stiffness is increased.

Chapter 6

An industrial example of substructuring

SUMMARY: This chapter describes the current state of the art research in the field of real-time dynamic substructuring. We demonstrate the broadening of the real-time technique by moving away from large scale seismic testing to 'small' scale component testing. The knowledge gained from the conceptually simple case studies is applied to a real engineering problem of a helicopter lag damper unit. An eight mode numerical model of a helicopter blade is excited using flight data and connected to the real damper unit via a 50kN actuator.

6.1 Introduction

A helicopter is an aircraft which is lifted and propelled by one or more large horizontal rotors (propellers). Helicopters are classified as rotary-wing aircraft to distinguish them from conventional fixed-wing aircraft. The word helicopter is derived from the Greek words *helix* (spiral) and *pteron* (wing). The engine-driven helicopter was invented by the Slovak inventor Jan Bahyl. The first stable, fully-controllable helicopter placed in production was invented by Igor Sikorsky.

Compared to conventional fixed-wing aircraft, helicopters are much more complex, more expensive to buy and operate, relatively slow, have poor range and restricted payload. The compensating advantage is maneuverability; helicopters can hover in place, reverse, and above all take off and land vertically. Subject only to refuelling facilities and load/altitude limitations, a helicopter can travel to any location, and land anywhere with a clearing a rotor disk and a half in diameter.

A conventional aircraft is able to fly because the forward motion of its angled wings forces air downwards, creating an opposite reaction called lift that forces the wings upwards. A helicopter uses exactly the same method, except that instead of moving the entire aircraft, only the wings themselves are moved. The helicopter's rotor can simply be regarded as rotating wings. Turning the rotor generates lift but it also applies a reverse force to the vehicle, which tries to spin the helicopter fuselage in the opposite direction to the rotor, this is called *torque reaction.* There are several possible design layouts for overcoming this problem, the most common design is the Sikorsky-layout (used by approximately 95% of all helicopters manufactured to date) — using a smaller vertical propeller mounted at the rear of the aircraft called a tail rotor.

Useful flight requires that an aircraft be controlled in all three dimensions. In a fixed-wing aircraft, this is "easy"; small movable surfaces are adjusted to change the aircraft's shape so that the air rushing past pushes it in the desired direction. In a helicopter, however, there often is not enough airspeed for this method to be practical. For rotation about the vertical axis, *yaw*, the pitch of the tail rotor alters the sideways thrust produced (dual-rotor helicopters have a differential between the two rotor transmissions that can be adjusted by an electric or hydraulic motor to transmit differential torque and thus turn the helicopter). For *pitch* (tilting forward and back) or *roll* (tilting sideways) the angle of attack of the main rotor blades is altered or *cycled* during the rotation creating a differential of lift at different points of the rotary wing. More lift at the rear of the rotary wing will cause the aircraft to pitch forward, an increase on the left will cause a roll to the right and so on.

In the early days of helicopter development, there were six fundamental problems:

1. Understanding the basic aerodynamics of vertical flight.
2. The lack of a suitable powerplant (engine).
3. Minimising structural weight and engine weight.
4. Counteracting rotor torque reaction.
5. Providing stability and properly controlling the machine.
6. Conquering the problem of high vibrations.

Over the last forty years, sustained scientific research and development in many different aeronautical disciplines has allowed for large increases in helicopter performance, lifting capability of the main rotor, high speed cruise efficiencies, and mechanical reliability. Continuous aerodynamic improvements to the efficiency of the rotor have allowed the helicopter to lift more than its empty weight and to fly in level flight at speeds in excess of 400 km/h. The improved design of the helicopter and the increasing viability of other vertical lift aircraft such as the tilt-rotor continue to advance as a result of the revolution in computer-aided design and manufacturing and the advent of new lightweight composite materials. However, since the 1980s, there has been an accelerating scientific effort to understand and overcome some of the most difficult technical problems associated with helicopter flight, particularly in regard to aerodynamic limitations imposed by the main rotor, points 5-6.

In this chapter, we discuss the origins of some of these highly complex problems and show how we can use real-time dynamic substructuring to better understand them in order to potentially reduce or remove them completely. We focus on one main feature, the *lag damper*, which is found on all *fully-articulated* helicopters. This damper is critical component with respect to the stability of the aircraft by controlling blade motion in a way to avoid resonances. Similarly, as a component

of the rotor hub dynamic system, the damper influences the general vibration characteristics of the entire aircraft. Kinematically, the damper helps control and react loads generated by blade lead-lag motion. The damper loads are typically transferred back into the airframe through the rotor hub where the "fixed end" of the damper is attached. Until now, only numerical simulations of how these dampers affect the helicopter's rotor dynamics have been utilized. Initially, these were linearized or simplified models of the damper's dominant characteristic behaviour. Recently, this work has been extended by Eyres [2] who developed a full parametric model of the damper excited by recorded flight data in order to try and capture its nonlinear behaviour. While this provides a far more accurate understanding of the damper's performance it is still only a numerical simulation with only a limited ability to validate its accuracy. The work presented in this chapter builds on this basis, using a flight certified lag damper as the experimental substructure and a real-time version of the blade dynamics as the numerical model (excited by the same flight test data).

6.2 Helicopter dynamic issues

Fixed wing aircraft are designed to be inherently stable. If a gust of wind or a nudge to one of the controls causes a fixed wing aircraft to pitch, roll, or yaw, the aerodynamic design of the aircraft will tend to correct the motion, and the aircraft will return to its original attitude. A small, fixed wing aircraft can be stable enough that a pilot can let go of the controls while looking at a map or dealing with a radio for example and the plane will generally stay on course. In contrast, helicopters are highly unstable. Simply hovering requires continuous, active corrections from the pilot. When a hovering helicopter is nudged in one direction by a gust of wind, it will tend to continue in that direction, and the pilot must adjust the specific control to correct the motion.

Adjusting one flight control on a helicopter almost always has an effect that requires an adjustment of the other controls. Moving the *cyclic* (the control column) forward

causes the helicopter to move forward, but will also cause a reduction in lift, which will require extra *collective* for more lift. Increasing the collective will reduce rotor angular velocity (RPM), requiring an increase in *throttle* to maintain constant rotor RPM (all helicopters are specifically designed to operate at a constant RPM). Changing the collective will also cause a change in torque, which will require the pilot to adjust the *foot pedals*.

The remainder of this section will outline the dynamic issues that lead to the inclusion of a mechanical damper on the main rotor blade of an articulated helicopter.

6.2.1 Unbalanced lift

As a helicopter moves forward, the rotor blade on one side moves at rotor tip speed plus the aircraft speed and is called the *advancing* blade. As the blade swings to the other side of the helicopter, it moves at rotor tip speed minus aircraft speed and

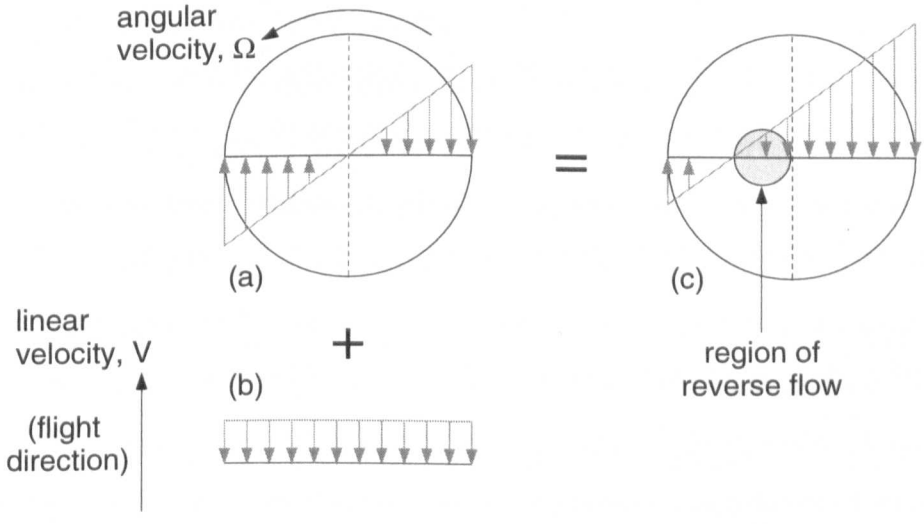


Figure 6.1: Flow velocity vectors on a rotor in forward flight as seen from above. (a) Flow due to rotating blades in position perpendicular to flight direction; (b) Flow over blades due to forward flight; (c) Combination of rotating motion and forward flight flow vectors.

is called the *retreating* blade. This is illustrated by the flow velocity vectors shown in Figure 6.1. The superimposed flow on the retreating side leads to a region of the rotor where the flow over the blades is in the opposite sense to the rotation of the blade — a reverse flow region.

To compensate for the added lift on the advancing blade and the decreased lift on the retreating blade, the rotor is allowed to *flap*. By allowing the advancing blade to flap upward, and the retreating blade to flap downward, it changes the angle of incidence on both rotor blades which balances out the entire rotor system. There are a few ways to allow for blade flapping. On fully-articulated rotor systems, the blades are allowed to flap on hinges, whereas on semi-rigid rotor systems the whole hub swings up and down around an internal bearing called a trunnion.

6.2.2 Fully articulated helicopters

Fully articulated rotor systems allow each blade to *feather* (rotate about the pitch axis to change lift), *lead and lag* (move back and forth in-plane), and *flap* (move up and down about an inboard mounted hinge) independent of the other blades. Fully articulated rotor systems are found on rotor systems with more than two blades. As the rotor spins, each blade responds to inputs from the control system to enable aircraft control. The centre of lift on the whole rotor system moves in response to these inputs to effect pitch, roll, and upward motion. The magnitude of this lift force is based on the collective input, which changes pitch on all blades in the same direction at the same time. The location of this lift force is based on the pitch and roll inputs from the pilot. Therefore, the feathering angle of each blade (proportional to its own lifting force) changes as it rotates with the rotor, thus is called "cyclic control".

As the lift on a given blade increases, it will want to flap upwards. The *flapping hinge* for the blade permits this motion, and is balanced by the centrifugal force of the rotating blade, which tries to keep it in the horizontal plane. However,

some motion must be accommodated. The centrifugal force is nominally constant, however the flapping force will be affected by the severity of the manoeuvre (rate of climb, forward speed, aircraft gross weight).

As the blade flaps, its centre of mass moves closer in plane to the hub which introduces large Coriolis forces[1]. A blade can be modelled simply as a particle at the blades centre of mass rotating about a central vertical axis at the blade root. When the blade flaps the effect would be to accelerate the particle upwards and move the centre of mass toward the axis of rotation relative to the horizontal plane. There must therefore be an acceleration (resulting in the Coriolis force) in the plane of rotation if momentum is to be conserved. If the blade were rigid in this horizontal plane then there would be a large force generated at the hub causing vibration and possible damage to the blades or attachments. To relieve this force a second hinge can be introduced to the blade, in a similar way to the flapping hinge, to reduce the moment at the hub. The blade is then allowed to move in the plane of the rotor disc, otherwise known as the lead-lag plane, so the hinge is called the *lag hinge*. However, in larger helicopters a mechanism to absorb and control this movement is required. Typically, this is done using a hydraulic damper to limit the resultant movement, see § 6.2.3 and § 6.3. Any time that some blades are flapping up higher than others, the centre of gravity of the rotor head will not align with its true physical axis. This will cause vibration, as forces acting on the rotor mass try to find a more acceptable axis point; this is called the Hookes Joint Effect. Additionally, A third hinge is included to allow the blades to change pitch, called the *feathering hinge* and is controlled by the main pilot controls by means of a pitch horn. It should be noted that all three degrees of freedom are coupled.

[1]The Coriolis forces is named after Gaspard-Gustave Coriolis, who introduced the concept in 1835 to allow Newton's laws of motion to be applied in the context of rotating bodies.

6.2.3 Inclusion of a lag hinge

The inclusion of the lag hinge solves the problem of the Coriolis forces being transmitted to the hub. It does however introduce problems of its own due to the lack of natural damping in the lag (rotor disc) plane. As stated in § 6.2.2, the flapping motion is inherently stable due to the centrifugal force of the rotating blade. This is not the case with the lead-lag plane because the only force on the blade in the lag plane is the aerodynamic drag of the blade, which is small compared to the other forces on the blade. Thus the inclusion of a mechanical damping device is required. There are many forms a lag damper can take, including constant force, elastomeric, variable orifice and controllable fluid dampers.

Without a lag damper when the helicopter starts up and the blades increase in rotational frequency, they pass through a range of angular velocities, some of which can cause problems if they are allowed to resonate with the fuselage. The situation can arise where the blades bunch together as they rotate. As a result the centre of mass of the helicopter will move in phase with the rotational speed of the rotor. In the worst-case scenario, the rotation of the blades can become in phase with the natural frequency of the helicopter — the lag mode becomes in phase with the fuselage and/or landing gear. The oscillations of the blades can increase in magnitude rapidly up to the point where the helicopter can topple over due to the movement of the centre of mass. This condition is known as *ground resonance* because the critical mode is typically an oscillation of the fuselage on the landing gear while the tyres are touching the ground [111]. Additionally, ground resonance is also a hazardous condition during touchdown. A series of shocks to the landing gear can pass through to the rotor disk and cause an imbalance in the rotor system. Under extreme conditions, the imbalance causes violent oscillations that quickly build and result in catastrophic damage of the entire airframe. In some cases, complete destruction occurs, e.g. body panels, fuel tanks, and engines are all ripped from their mountings.

There is an additional resonant condition when the helicopter is in flight called *air resonance.* In this case, the fuselage is no longer rigid and the problem occurs when the cyclic lag mode becomes in phase with the coupled flap modes and fuselage modes of vibration, however, fully articulated helicopters tend not experience air resonance [112].

As with the inclusion of a lag hinge, the lag damper introduces problems of its own. Although the lag damper can damp out some additional undesirable interactions and vibrations during flight, it is a closed system and absorbs a large amount of energy which is dissipated as heat. Therefore, the passive lag damper actually detracts from the helicopter performance during steady state flight but is essential for take-off and landing so cannot be removed.

6.3 The EH101 lag damper

Early helicopters used frictional lag dampers to prevent ground resonance. A friction damper simply attempts to generate a constant force opposing the motion of the blade which is independent of velocity and frequency. However, the effective damping of the friction damper decreases with an increase in lag motion [113]. While the damping is acceptable for small amplitude lag motion, at higher amplitudes the damper does not generate enough force. This can be overcome by using multiple layers of various frictional materials. However, this adds to the complexity of the device especially as the damping force is dependant on the ambient temperature and humidity. Additionally, there is a possibility that the blades will stick when changing direction due to the discontinuity at zero velocity. This can cause a severe oscillation in the blade and potentially cause significantly damage the helicopter.

To overcome these problems, hydraulic lag dampers were developed in the 1960's that created a force proportional to the square of the lag velocity by forcing fluid through an orifice. The damper studied here is one of the latest generations of such a viscous fluid damper; it is a flight spec lag damper off an AgustaWestland

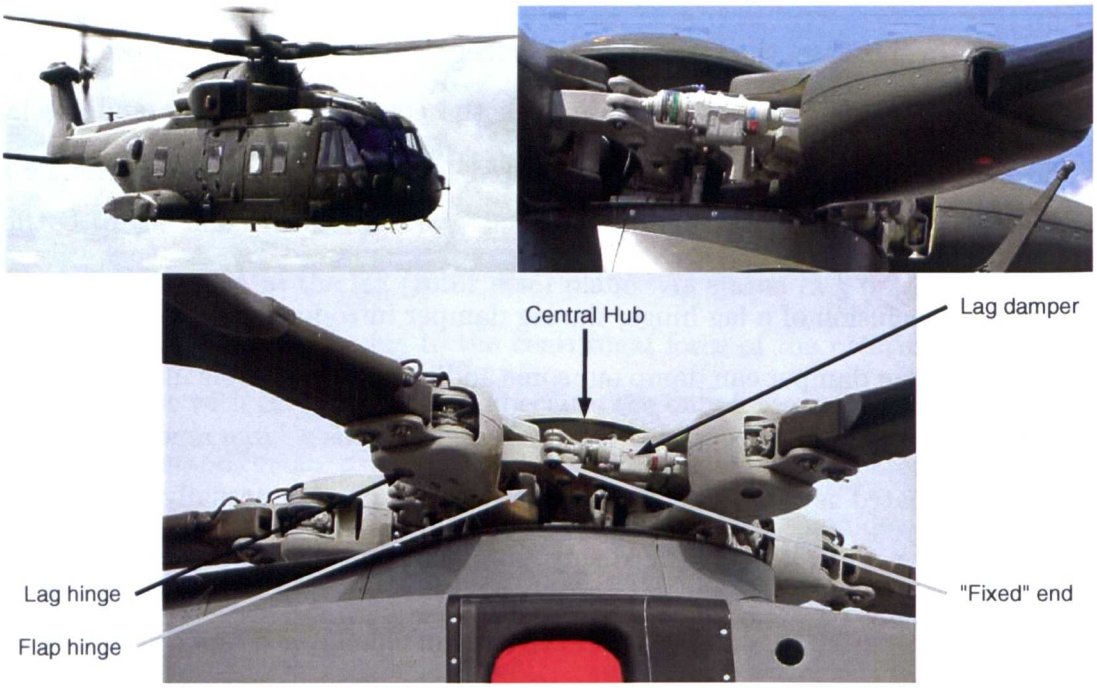


Figure 6.2: Close-up of the EH101 hub rotor system.

EH101 helicopter[2]. Figure 6.2 shows the lag damper in situ on the helicopter rotor hub. The main body of the damper consists of a cylindrical sealed chamber with a piston and rod passing through it. The damping force is generated by creating a pressure difference between the two sides of the piston. As the piston rod moves through the chamber, fluid is forced past the piston to the other side either round the outside of the piston or through the orifice itself. However, it is assumed that the only flow around the sides of the piston is for lubrication purposes and is negligible compared to the flow through the pistons orifice. In this way, a steep force-velocity characteristic can be generated for low velocities to approximate the

[2]The AgustaWestland EH101 is a medium-lift helicopter originally developed as a joint venture between Westland Helicopters in the UK and Agusta in Italy for military applications but also marketed for civil use. In 2001 AgustaWestland signed a deal with Lockheed Martin to market the aircraft in the US under the designation US101. It won the bid for the VIP and "Marine One" Presidential transportation (roles currently carried out by H-3 Sea King or the smaller UH-60 Black Hawk).

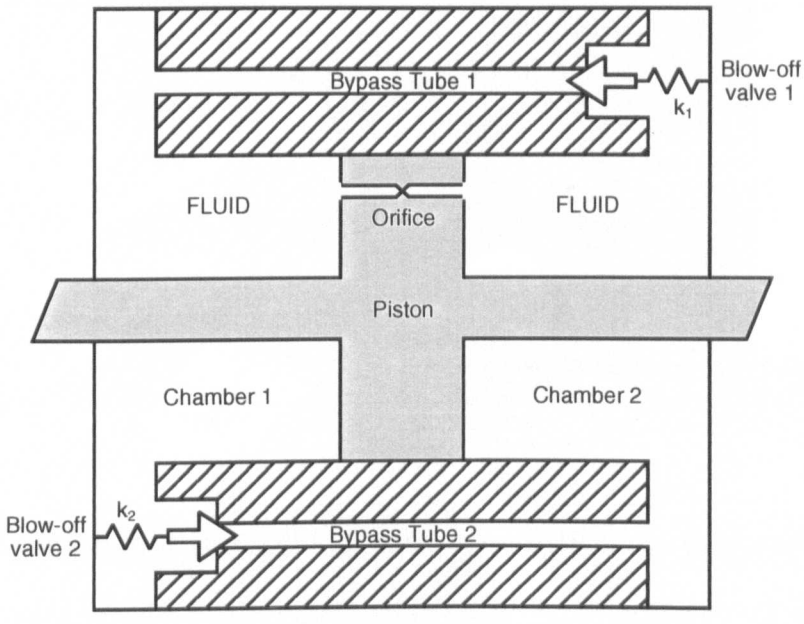


Figure 6.3: Cross-section of the hydraulic lag damper, including the relief valve orientations — Adapted form Eyres [2].

change in force of the friction damper as the blade velocity changes sign. The characteristic is then continuous at zero velocity and so removes the possibility of the blade sticking. Hydraulic dampers require *relief valves* to produce a useful force-velocity characteristic. The EH101 lag damper makes use of two valves connected to the damper casing and are operated by linear springs, one for each direction of motion of the piston. The precompression of the spring defines the force or pressure at which the valve will open. The valve dimensions and characteristics will determine the force-velocity gradient when the relief valve is open. A schematic of the damper modelled is shown in Figure 6.3.

The resulting difference in pressure between the two sides of the piston generates a force on the piston and hence the rod. In the case of the lag damper the motion of the blade relative to the hub will cause the damper rod to move through the damper casing. The force generated by the damper will act on the blade in the opposite

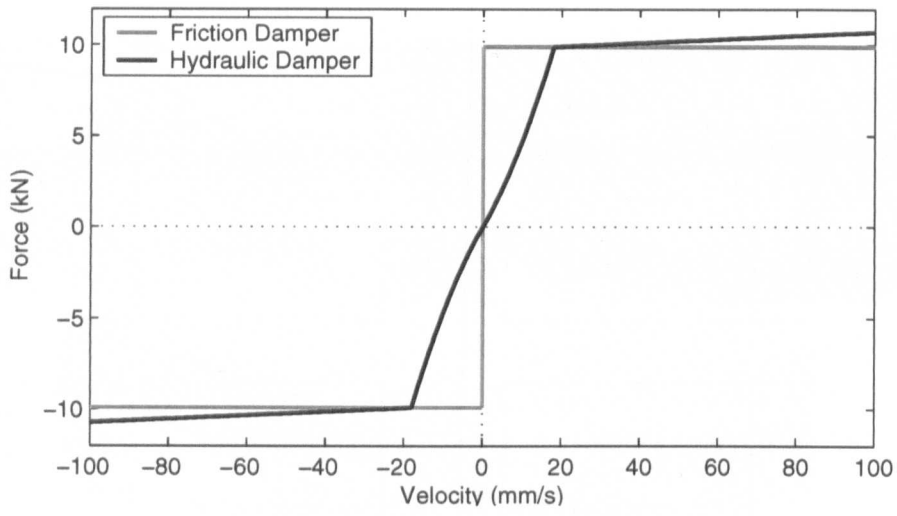


Figure 6.4: Comparison of the damping characteristics of an idealized friction damper and an idealized hydraulic damper with relief valves

sense to the motion, reducing the motion of the blade in its lag degree of freedom. The damper is attached to the blade and hub using universal joints. The force on the rod is thus assumed to act purely along the axis of the rod.

A schematic comparison between the force-velocity profiles of an idealized friction damper and an idealized hydraulic damper can be seen in Figure 6.4. It should be noted that the idealized characteristic behaviour of the hydraulic damper does not take into account all the nonlinear nature of the fluid or the spring-loaded relief valve dynamics as will be seen in § 6.4.1. In fact, it is this complex behaviour which makes the damper so hard to model numerically.

6.4 Experimental set-up

The constituent parts of the experimental set up are similar to that described in § 2.3.3, however, in order to achieve the performance required it has been necessary to significantly upgrade each of them. Figure 6.5 shows the experimental test rig setup including the EH101 lag damper. A standard size "hard hat" has been introduced for scale, in conjunction with Figure 6.2 (where we see the lag damper

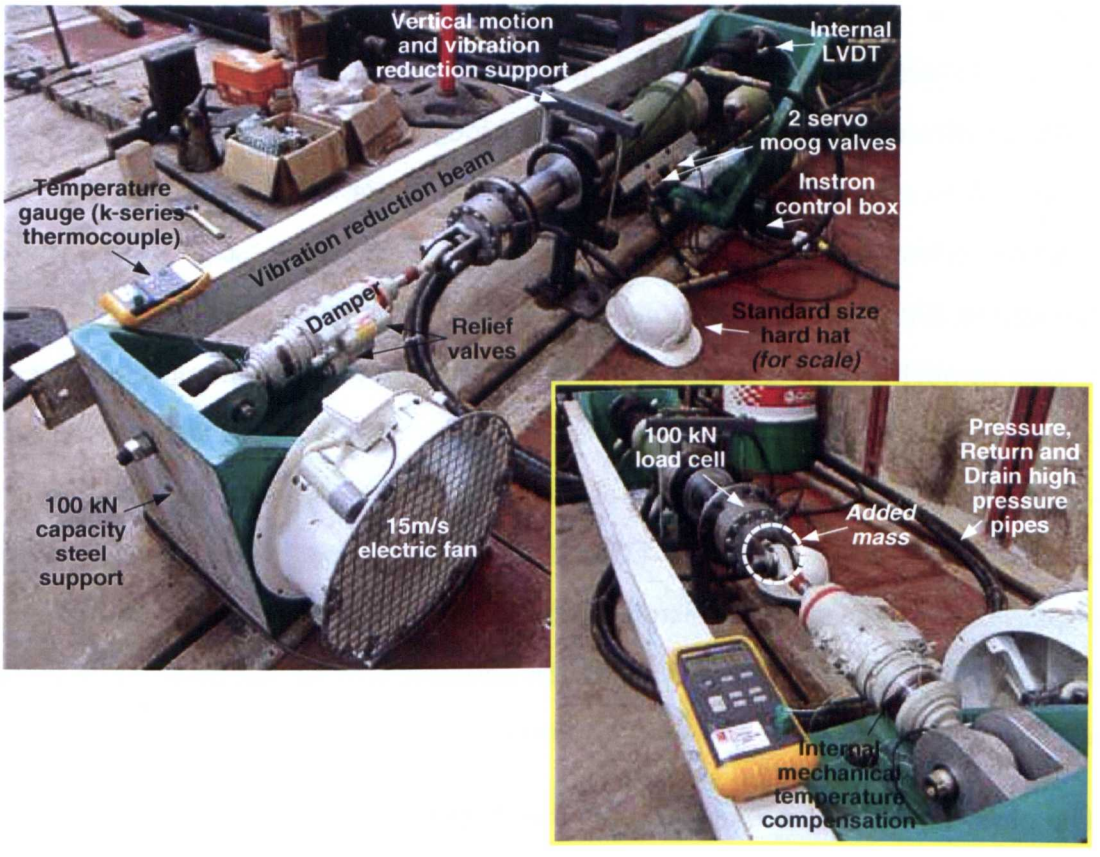


Figure 6.5: Experimental test rig setup for the EH101 lag damper; Note, the standard size "hard hat" for scale.

in situ) the reader can gain some understanding of the magnitude and capacity of the EH101 helicopter and therefore the damping forces which must be achieved by the lag damper.

Using flight data and information gained from previous investigations into the damper [2], the operational performance criterion for steady state flight at 84knots is a magnitude of 15kN of damping force required at a maximum velocity of approximately 300mm/s. It is a fundamental characteristic that there is drop off in load capacity of an actuator as the velocity of the piston is increased, as discussed in § 4.3.7 (Figure 4.15). Also in that section, we discussed the relationship between the local and global error in terms of a substructuring algorithm. In order to keep the global error as close to the local error as possible (i.e. achieve the most accurate test

possible), the actuator should not be driven far in excess of 75% of its capacity. In order to achieve the requirements for these tests, it has been necessary to use a 50kN hydraulic actuator with two servo Moog valves (in parallel) as shown in Figure 6.5 to achieve a maximum velocity of 500mm/s. The actuator is plumbed into the hydraulic ring main which can deliver a capacity of 486 l/min of oil at a pressure of 200 bar to the lab (only 100 l/min of oil is required for the test). This is done through *pressure*, *return* and *drain* high pressure hydraulic pipes connected to an accumulator, which in turn is connected to the ring main. As this is only a single actuator experiment only one channel is required on the accumulator. An internal LVDT (Linear Variable Differential Transformer) is used to measure the displacement of the actuator piston which has a $\pm 0.01\%$ linearity error on full scale deflection of 140mm. The control of the actuator will be discussed later in the section.

The physical rig itself is quite simple. Two 100kN steel supports (painted green in Figure 6.5) are bolted directly into a steel T-slot in the "strong" floor of the lab with the central axis of the actuator and damper aligned. The base of the actuator is rigidly bolted into one support and then supported by a vertical stand. This stand has a height adjustment feature allowing for alignment and additionally ensures that the actuator does not vibrate during testing. A $\pm$100kN Instron Dynacell dynamic load cell is then rigidly attached to the actuator piston. A yoke is required to connect the damper to the active part of the load cell. This extra mass is then included within the substructure and can skew the inertial response of the substructure, in this case the lag damper piston, such that $F_{cell} = F_{damper} + m_{added}a_{piston}$. The yoke can be seen in Figure 6.5 and is labelled as *added mass*. The Instron Dynacell is a load measurement device which automatically compensates for load errors induced through inertia by automatically tuning a compensation factor k_{lc} which is used in conjunction with an internal axially mounted accelerometer a_{lc}. Thus,

$$F_{cell} = F_{damper} + m_{added}a_{piston} - k_{lc}a_{lc} = F_{damper}. \tag{6.1}$$

The dynamic inertia compensation is essential for real-time dynamic substructure

testing. The lag damper is fitted either end with aircraft grade universal joints. These allow deviation in all directions while being manufactured under extremely high tolerances such that axial motion is eliminated. The relief valves are orientated towards the air flow generated by an electric fan, which produces an air flow of nominally 15m/s (replicating the environment in which the damper would operate in practice), in order to achieve the greatest amount of cooling. The damper is a closed system and as such expels any energy generated as heat. The damper is designed to operate at between ambient and 50°C in normal flight conditions, increasing to 80°C in desert conditions. The oil seals fail at 120°C. To keep the viscosity constant within the damper during operation (i.e. during changes in temperature) an internal mechanical spring-loaded compensator is integral to the damper's design. In order to observe the temperature change during testing a K-series thermocouple has been attached to the outer casing of the damper and is read on a digital multimeter — this is not used for any control, just to ensure the correct testing environment.

The base end of the lag damper is then bolted into a yoke directly attached to the remaining steel support. Finally, a 5 inch steel channel section is bolted directly onto the steel supports under tension. This preloads the rig which helps to remove any vibration and unwanted axial displacement. Under test conditions the unwanted axial displacement was measured at $\pm 0.1mm$ over the entire length of the rig setup. Final fine scale alignment for the entire rig was carried out using a theodolite and a laser projection system.

The control of the test rig is achieved in a similar manner to that described in § 2.3.3, but again upgraded to achieve the desired performance requirements. The control station is shown in Figure 6.6 and consists of four constituent components:

Instron 8800 Control Hardware Two Instron 8800 Tower Controllers are present in the setup allowing for a fully integrated eight-channel control for any general purpose multi-axis hydraulic test application. However, for this experiment we only require one channel for the single actuator. These hardware controllers

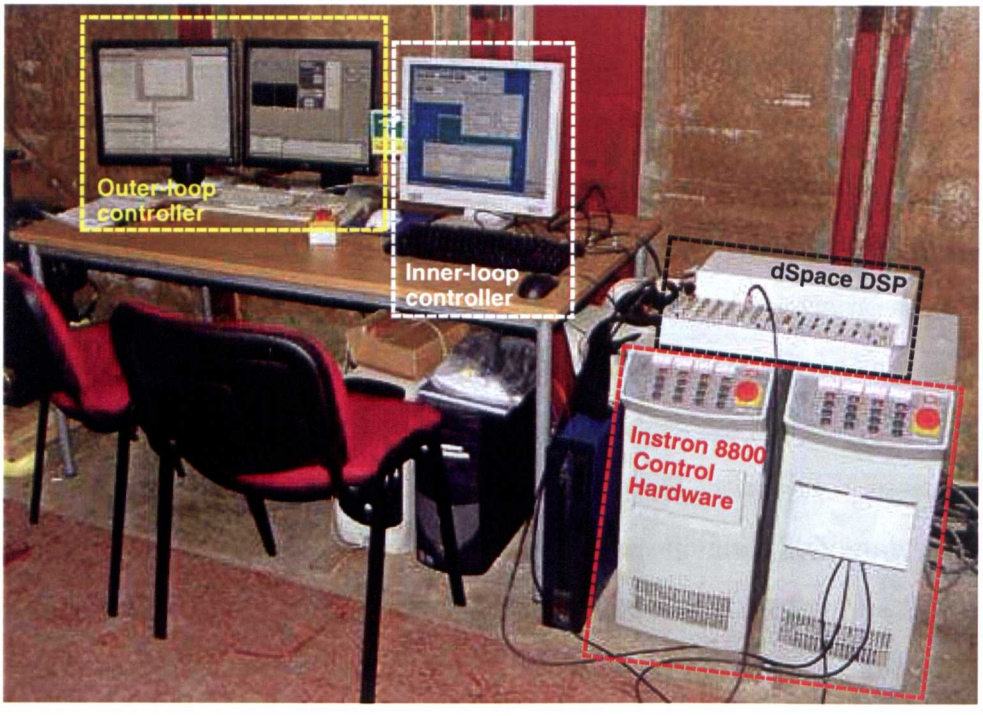


Figure 6.6: Experimental test rig controller station; 4 components: 8 channel Instron 8800 hardware control system, dSpace 1103 DSP processor and expansion board, Inner-loop control computer and Outer-loop/substructuring algorithm computer.

are state of the art digital servohydraulic controllers which have 19 bit data resolution across the entire span of each transducer in order to provide the maximum data quality. Additionally, the controller offers automatic identification and calibration of all compatible transducers for accuracy and reliability.

Inner-loop controller Rather than use the Instron 8800 controllers as an analogue amplifier (in a similar way to how the Panasonic AC servo motors were configured for the case study experiments) a standalone PC is used for the *inner-loop* linear PID control of the transfer system. The primary reason for this is safety, due to the increased safety risk should anything go wrong. While all the inner-loop control could be handled by the dSpace architecture (as is done for the case studies) external safety limits for each transducer and for

the hydraulic system can be directly set and monitored by this PC as it is integrated with the Instron control towers through dedicated software.

Outer-loop controller A second standalone PC is used for the substructuring algorithm and *outer-loop* control. As before, the substructuring algorithm is designed in MATLAB/Simulink (obtaining the benefits of working on a general purpose machine) before being compiled and then built into the hard real-time processor of the dSpace DSP. Additionally, online visualization and data acquisition is controlled through this PC using the dSpace companion software ControlDesk. From here we can control the parameters of the substructuring algorithm such that a variety of testing applications can be performed.

dSpace DSP The dSpace DS1103 R&D Controller Board is used to implement the substructuring algorithm experimentally in real-time. The substructuring algorithm is built into the processor (which operates at a clock speed of 500 MHz) and is connected to the Instron controller via an expansion board. The relevant signals are then passed between the Instron controller and the dSpace DSP under hard real-time constraints. The dSpace board outputs to the outer-loop PC in soft real-time for visualization.

Essentially, the inner-loop controller and Instron control tower are used to activate the system, achieve a high quality, repeatable response from the actuator (such that the transfer system has low uncertainty) and then to monitor the experimental signals during operation to ensure that everything is operating within the correct physical tolerances. The tests are then completely controlled from the outer-loop controller PC which simply changes the parameters values within the dSpace model, which in turn provides the demand signal to the activated inner-loop system.

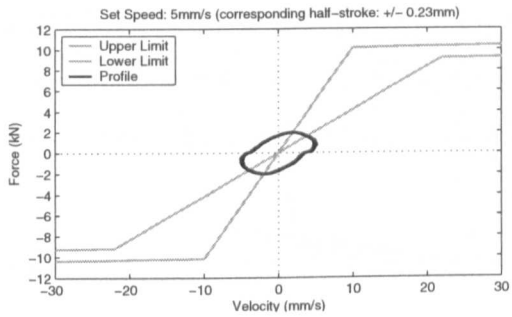
6.4.1 Lag damper system identification

It is important to carry out a full system identification of the damper in order to understand why real-time dynamic substructuring of this component can be of such value. Table 6.1 shows the dynamic testing points set for the EH101 damper as prescribed by Westland Helicopters Ltd. All excitation is sinusoidal at a frequency of 3.5Hz (that of the rotor system in flight). The requirements state that the amplitude of the measured signal should not exceed ±5% of the nominal value. For this reason, the amplitude correction σ of the AFP algorithm was used in isolation for these tests, see § 4.3.6.

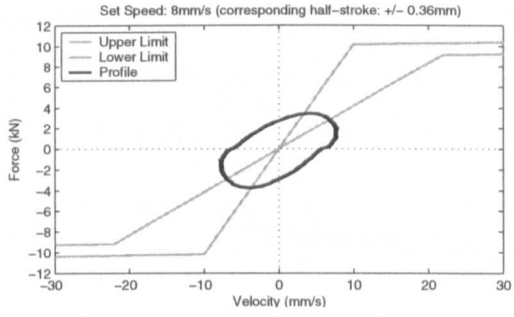
Figures 6.7 and 6.8 show the corresponding experimental *force-velocity* profiles produced by the damper for the differing set speed conditions as described in

Test No.	Set Speed (mm/s)	Corresponding half-stroke (mm)
1	5	± 0.23
2	8	± 0.36
3	10	± 0.45
4	14	± 0.64
5	18	± 0.82
6	21	± 0.95
7	25	± 1.14
8	130	± 5.91
9	200	± 9.09
10	300	± 13.64
11	450	± 20.46
12	600	± 27.28

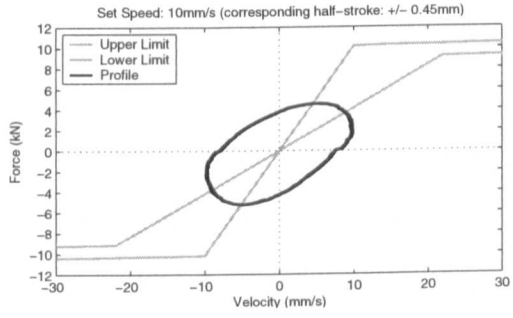
Table 6.1: Dynamic testing points set for the EH101 damper as prescribed by Westland Helicopters Ltd.



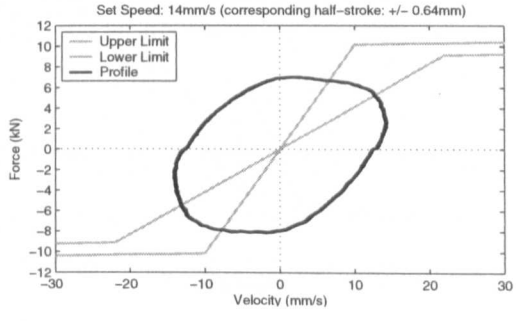
(a) Test No. 1



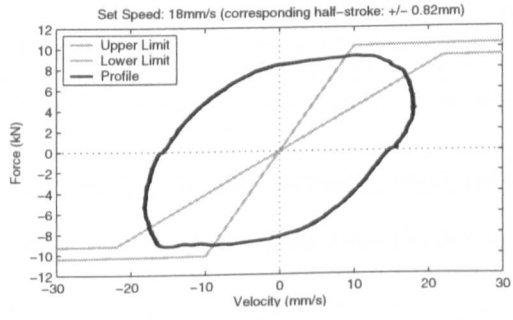
(b) Test No. 2



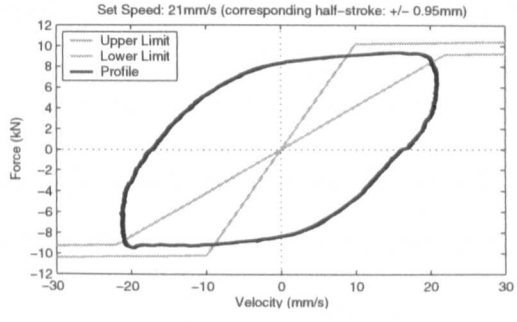
(c) Test No. 3



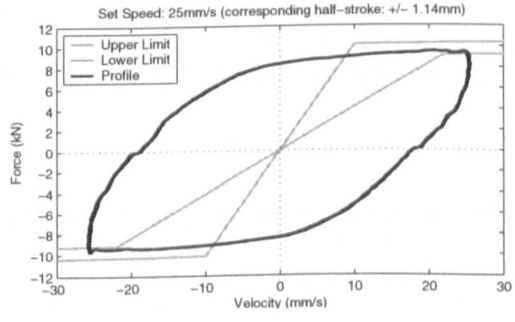
(d) Test No. 4



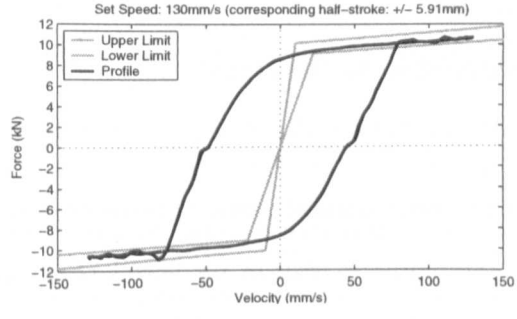
(e) Test No. 5



(f) Test No. 6

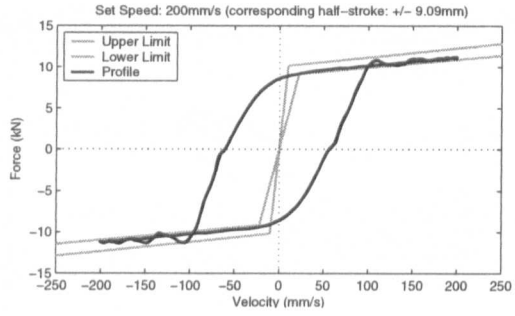


(g) Test No. 7

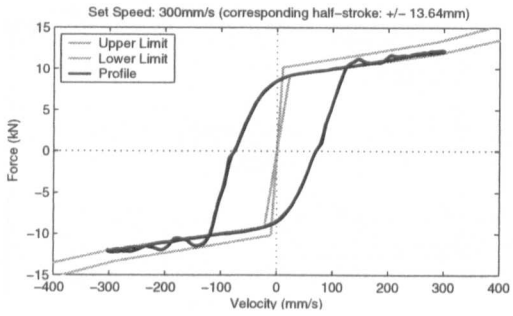


(h) Test No. 8

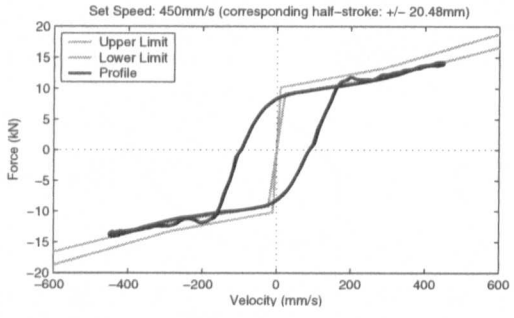
Figure 6.7: Table 6.1 (1-8): Experimental Force-Velocity Profile.



(a) Test No. 9



(b) Test No. 10



(c) Test No. 11

Figure 6.8: Table 6.1 (9 - 11): Experimental Force-Velocity Profile. Maximum velocity of actuator is rated at 500mm/s; therefore, Test No.12 cannot be performed.

Table 6.1 for 5 seconds worth of steady state data (orientation of profile is anti-clockwise). Note that the experimental velocity signal has been filtered in a post-process procedure using a 4^{th} order Savitzky-Golay (S-G) smoothing filter with a vector size of 21. S-G filters are optimal in the sense that they minimize the least-squares error in fitting a polynomial to frames of noisy data without introducing a phase shift to the signal. The S-G filter is not suitable for online use as it requires data in front and behind the current time step.

The experimental data is superimposed over the manufacture's upper and lower tolerances. It should be noted that the entire profile is not designed to fit within these tolerances, instead just the peak position. This highlights one of the major limitations in the understanding of the damper's dynamic characteristics as when designing the damper, it is just these tolerances which are specified.

The most obvious experimental characteristic of an actual lag damper compared

to that of the idealized profile of Figure 6.4 is its hysteretic behaviour after the piston changes direction. The extent of this is controlled by the size of the orifice in the piston and the viscosity/compressibility of the oil, all of which are fixed after manufacture/assembly for this type of passive damper. A further significant nonlinearity is seen as soon as the relief valve opens, this is most clear in Tests 8-11. The oscillation that is observed is the relief valve spring "bouncing". This is small in Figure 6.7(h) as the valve only opens for a short time as the damper piston is almost at the end of its cycle such that the acceleration is low. However, compared to Figure 6.8(c), when the valve opens while the damper is being driven at a far higher acceleration, we now observe sizeable nonlinear oscillations. The shape and magnitude of this nonlinearity is repeatable for each set condition, as 5 seconds of data is shown for each test, but is different for each test. Therefore, when the damper is not being driven in this simple manner, such as in-flight, it becomes increasingly complicated to model these dynamics. The combination of these two nonlinearities, and the fact that they vary with the operational environment of the damper, have made numerical modelling of such dampers extremely difficult.

It is noted that the slight nonlinearity at zero force is the actuator dead zone (a certain pressure is required to overcome the static friction of the piston). However, this nonlinearity is markedly better than the servo mechanical actuators of the case studies can achieve.

6.5 Scope of the current work

As previously stated, until recently only linearized or simplified computer simulations of a helicopter lag damper dynamics have been used. Eyres [2] extended this work by developing a full parametric model of the damper excited by recorded fight data in order to try and capture its nonlinear behaviour, as discussed in the previous section. While this provides a far more accurate understanding of the damper's performance it is still only a numerical simulation with only a limited

ability to validate its accuracy.

PsD substructuring has achieved a high level of success in the field of earthquake engineering. Real-time dynamic substructuring, on the other hand, is still very new in terms of any commercial applications. Here, we present work which was carried out in collaboration with Westland Helicopters Ltd. as a feasibility study to show how the testing technique can be applied to such a complicated mechanical problem. Westland Helicopters Ltd. believe that this could be the basis for future design of a more efficient lag damper. This application represents a broadening of the real-time substructuring technique from the large civil structural engineering applications to also being viewed as a valid smaller scale component testing technique.

In this chapter we build on the work of Eyres [2] to create a real-time dynamic substructuring algorithm of a numerical model of a single blade being excited by steady-state flight data connected to a flight certified lag damper. This is not a part step to demonstrating the effectiveness of substructuring as a dynamic testing technique, but a complete validation of the experimental techniques and strategies presented in this thesis. The future direction of the research (see § 7.2) will lie in creating more involved numerical models of a coupled five bladed rotor system under changing flight conditions or in developing smart damping devices. However, essentially the same tests would be being performed as will be described for this pilot study.

6.6 A simplified model of a blade in rotation

The original rotor code (called R150 and supplied by Westland Helicopters Ltd.) models the motion of a single blade under one full revolution. The code iterates until the blade's motion becomes periodic at which point the helicopter is said to be in a *trim* condition. Physically this would occur when the helicopter is flying at a constant velocity, altitude and direction or constant manoeuvre for a sufficiently long period of time to remove transients. Additional features are included in the

model, such as trailing edge vortices, but the general characteristics are described by a modal model. The full code is based on work developed by NASA in the 1950's and 60's [114, 115].

A simplified version of the model is outlined in this section and was first presented in Eyres [2]. This derivation assumes the blades are forced periodically by a constant matrix, modified to take into account the new force from the experimental lag damper. The analysis uses 8 modes represented by ϕ_i where $i = 1 \ldots 8$. The modes correspond to 4 flap modes, 3 lag modes and one twist mode. A disadvantage of using a modal approach is that the lag damper model cannot be directly incorporated into the equations [116]. Instead, the forcing effect of the damper is included on the right hand side of the forced response equation, thus giving the equation of motion for a given mode to be

$$\frac{1}{\Omega^2}\frac{d^2\phi_i}{dt^2} + \frac{2\upsilon_i^B\lambda_i^B}{\Omega}\frac{d\phi_i}{dt} + (\lambda_i^B)^2\phi_i = \frac{1}{I_i^B}(MF_i^{code} - LDMF_i^{code} + LDMF_i^{exp}), \quad (6.2)$$

with modal frequency λ_i^B, modal inertia I_i^B and structural damping υ_i^B of the blade for an angular velocity of Ω. The set of equations are forced by the constant matrices MF_i^{code}, the total modal forcing from the main rotor code and the lag damper modal forcing, $LDMF_i^{code}$. These three modal forcing matrices are periodic functions of time (or azimuth). The damper forcing is removed from the total forcing and replaced by the modal forcing measure from the physical substructure, $LDMF_i^{exp}$, calculated in Eq. (6.21) which is now a function of the modes. This effective total modal forcing drives the set of differential equations in Eq. (6.2). The motion of the damper is defined by the coordinates in Figure 6.9.

The motion of the blade in the flap, lag and torsion planes combine to produce a motion of the blade relative to its fixed rotating position. The flapping and lagging angle of the blade is given as β^B and ζ^B respectively. The flap and lag angles are calculated by constant vectors ω_D^B (the modal flap deflection) and $\bar{v}_D^B$ (the modal lag deflection) multiplied by the current modal state ϕ such that

$$\beta^B = \omega_{0_D}^B + \sum_{i=1}^{8} \phi_i \omega_{i_D}^B, \quad (6.3)$$

$$\zeta^B = v_{0_D}^B + \sum_{i=1}^{8} \phi_i \bar{v}_{i_D}^B, \tag{6.4}$$

where, $\omega_{0_D}^B$ and $v_{0_D}^B$ are the initial values. The angle of twist, θ^B, is simply a sinusoidal function of the modal state

$$\theta^B = \theta_0^B + \frac{d\theta_0^B}{d\psi} - A_1^B \cos(\psi) - B_1^B \sin(\psi), \tag{6.5}$$

where, A_1^B and B_1^B are the *lateral* and *longitudinal* cyclic control angle constants respectively, θ_0^B is the initial angle and ψ is the azimuth angle. The angular velocities of the three motions can then be found as

$$\frac{d\beta^B}{d\psi} = \sum_{i=1}^{8} \frac{d\phi_i}{d\psi} \omega_{i_D}^B, \tag{6.6}$$

$$\frac{d\zeta^B}{d\psi} = \sum_{i=1}^{8} \frac{d\phi_i}{d\psi} \bar{v}_{i_D}^B, \tag{6.7}$$

$$\frac{d\theta^B}{d\psi} = A_1^B \sin(\psi) - B_1^B \cos(\psi). \tag{6.8}$$

Using Figure 6.9 the rotation matrices can be derived as follows. The global position at B and D is given by the vectors $\underline{\boldsymbol{b}}$ and $\underline{\boldsymbol{d}}$ respectively. The position and velocity of D in the global coordinates can then be expressed in terms of the blades relative motions in flap, lag and twist as

$$\underline{\boldsymbol{d}} = \underline{\boldsymbol{b}} + T_\beta T_\zeta T_\theta (\underline{\boldsymbol{d}} - \underline{\boldsymbol{b}}), \tag{6.9}$$

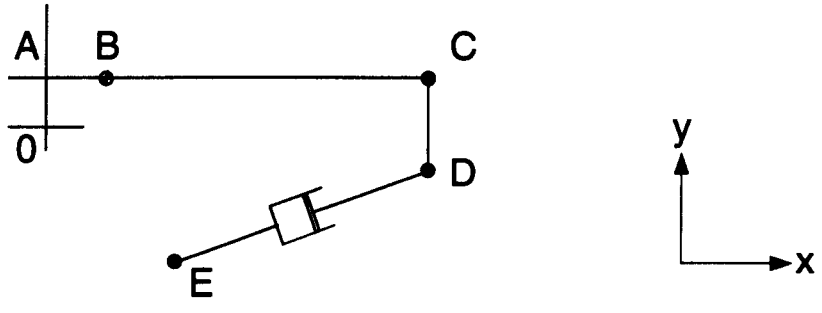


Figure 6.9: Geometry of how lag damper is attached to the blade: "0" represents the centre of the hub, B represents the flap hinge and C represents the lag hinge.

$$\dot{\underline{d}} = \dot{T}_\beta \dot{T}_\zeta \dot{T}_\theta (\underline{d} - \underline{b}), \tag{6.10}$$

where the rotation matrices for flap T_β, lag T_ζ and twist T_θ are given as

$$T_\beta = \begin{bmatrix} \cos(\beta^B) & 0 & -\sin(\beta^B) \\ 0 & 1 & 0 \\ \sin(\beta^B) & 0 & \cos(\beta^B) \end{bmatrix}, \tag{6.11}$$

$$T_\zeta = \begin{bmatrix} \cos(\zeta^B) & -\sin(\zeta^B) & 0 \\ \sin(\zeta^B) & \cos(\zeta^B) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{6.12}$$

$$T_\theta = \begin{bmatrix} 0 & 0 & 1 \\ 0 & \cos(\theta^B) & -\sin(\theta^B) \\ 0 & \sin(\theta^B) & \cos(\theta^B) \end{bmatrix}. \tag{6.13}$$

The velocities at D are transformed into the damper axis to give the velocity of the damper piston, V_d, using the fixed position $\underline{e}$ at E. Subscripts indicate the component of the vector that is being used. For example $\underline{e}_Z$ is the component of the position vector at E in the direction of the z-axis.

$$V_d = T_\gamma^{-1} T_\delta^{-1} \dot{d}, \tag{6.14}$$

where

$$T_\gamma = \begin{bmatrix} \cos(\gamma^B) & -\sin(\gamma^B) & 0 \\ \sin(\gamma^B) & \cos(\gamma^B) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{6.15}$$

$$T_\delta = \begin{bmatrix} \cos(\delta^B) & 0 & -\sin(\delta^B) \\ 0 & 1 & 0 \\ \sin(\delta^B) & 0 & \cos(\delta^B) \end{bmatrix}, \tag{6.16}$$

for the angles γ^B and δ^B representing the angles of the damper relative to the points D and the fixed point E on the hub. The angles are calculated as

$$\delta^B = \tan^{-1}\left[\frac{\underline{d}_Z - \underline{e}_Z}{\underline{d}_X - \underline{e}_X}\right], \tag{6.17}$$

$$\gamma^B = \cos^{-1}\left[\left[(\frac{\underline{d}_X - \underline{e}_X}{L_D})^2 + (\frac{\underline{d}_Z - \underline{e}_Z}{L_D})^2\right]^{\frac{1}{2}}\right], \tag{6.18}$$

where L_D is the absolute distance between the two attachment points of the lag damper expressed as

$$L_D = [(\underline{d}_X - \underline{e}_X)^2 + (\underline{d}_Y - \underline{e}_Y)^2 + (\underline{d}_Z - \underline{e}_Z)^2]^{\frac{1}{2}}. \tag{6.19}$$

The resulting velocity, V_d, used in the damper equations is the output from the substructuring numerical model.

The force measured from the damper, $\underline{F} = [A\Delta P, 0, 0]$ (where, A is the cross-sectional area of the piston and ΔP is the pressure difference between chambers 1 and 2), is transformed back into the global axis system at D to give $\underline{FD}$ so the modified forcing of the modes can be calculated as

$$\underline{FD} = T_\gamma T_\delta \underline{F}, \tag{6.20}$$

and at C using the fact that $\underline{FC} = \underline{FD} := [FC_X, FC_Y, FC_Z]$.

The modal forcing provided by the damper is given for the i^{th} mode as

$$LDMF_i^{exp} = \frac{1}{\Omega^2}\sum_{j=1}^{7} T_i^{(j)}, \tag{6.21}$$

where the seven quantities $T_i^{(j)}$ are calculated for each mode using small angle approximations and constant vectors ω_{i_D}, $\bar{v}_{i_D}$ and t_{i_D} for flap, lag and torsion such that

$$\begin{aligned}
T_i^{(1)} &= FC_X(-\theta^B(\underline{d} - \underline{b})_Y - (\underline{d} - \underline{b})_Z - (\underline{d} - \underline{b})_X\beta^B)\omega_{i_D},\\
T_i^{(2)} &= FC_X(\theta^B(\underline{d} - \underline{b})_Z - (\underline{d} - \underline{b})_Y - (\underline{d} - \underline{b})_X\zeta^B)\bar{v}_{i_D},\\
T_i^{(3)} &= FC_X(-\beta^B(\underline{d} - \underline{b})_Y\zeta^B(\underline{d} - \underline{b})_Z)t_{i_D},\\
T_i^{(4)} &= FC_Y\bar{v}_{i_D},\\
T_i^{(5)} &= -FC_Y(\underline{d} - \underline{b})_Z t_{i_D},\\
T_i^{(6)} &= FC_Z\omega_{i_D},\\
T_i^{(7)} &= FC_Z(\underline{d} - \underline{b})_Y t_{i_D}.
\end{aligned} \tag{6.22}$$

6.7 Numerical substructuring simulation of the idealized lag damper

The first stage of creating a substructuring algorithm is to convert the derivation, described in the previous section, into a format that can be compiled into the real-time processor of the dSpace DSP. Due to the complexity of the model required and the need for the code to run as efficiently as possible, the substructuring numerical model has been built into a real-time S-Function, as described in § 2.4.3. Refer to Appendix B for the actual S-Function code. Additionally, if we also develop a separate numerical model for the idealized substructure (the viscous damper shown in Figure 6.4) we can run a purely numerical substructuring test, effectively setting $\gamma = 0$ from § 5.3.3. As this model of the substructure will only be used for development purposes and for the robustness compensation (see § 6.9.1) an accurate model of the damper is not required, just one which has similar characteristics.

Although for the full experimental substructuring algorithm it will not be possible to model an emulated system, it is, in a sense, possible for our numerical-numerical substructuring algorithm. If we can have confidence in the numerical algorithm, it

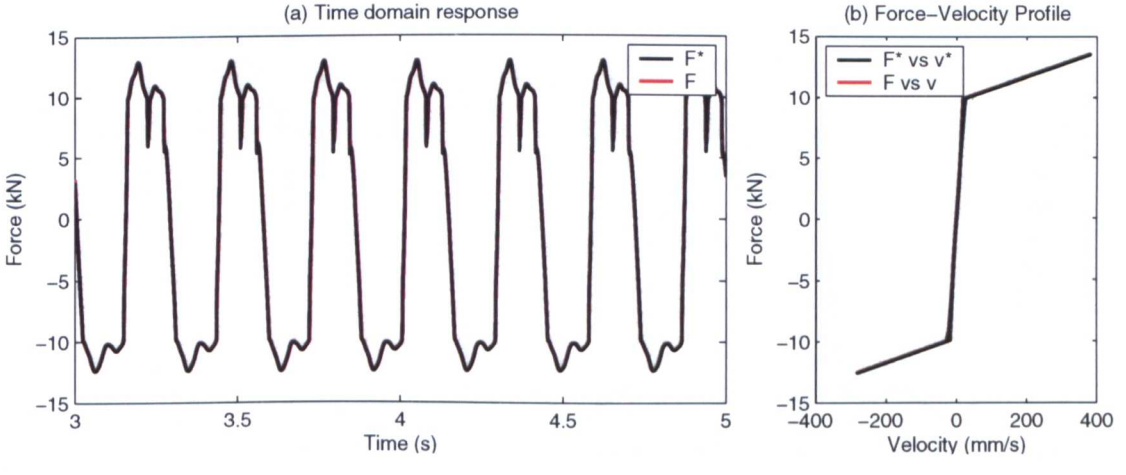


Figure 6.10: A comparison of the idealized Eyres [2] damper model [F* v*] excited by flight data to a continuous-time numerical-numerical substructuring algorithm test [F v] excited under the same conditions.

will allow us to draw the same conclusions about accuracy of the final experimental results as discussed in § 4.3.7. Using the same idealized lag damper model for both the numerical substructuring algorithm and the model designed by Eyres [2] and then exciting them both under the same flight conditions and from the same flight data we can compare their results. Figure 6.10 shows such a test for a flight speed of 84knots. We see an extremely high correlation in the time domain responses, Figure 6.10 (a), between the emulated system (the Eyres [2] model) and a continuous-time numerical-numerical substructuring algorithm. From Figure 6.10 (b) we observe identical force-velocity profiles for the modelled damper. We can take this result as a validation of the numerical substructuring algorithm.

However, the substructuring algorithm must operate in discrete-time when performing an experimental test. The computation duration can be identified from the system state feedback from the dSpace processor. For the 1103 board, the minimum *turnaround time* to operate the substructuring algorithm is 0.82ms. If we repeat the previous test at a discrete sample rate of 1kHz we can see the effect on the results, as shown by Figure 6.11. It is clear that the discretization of the force signal from the numerical substructure can excite some of the high frequency modes when the

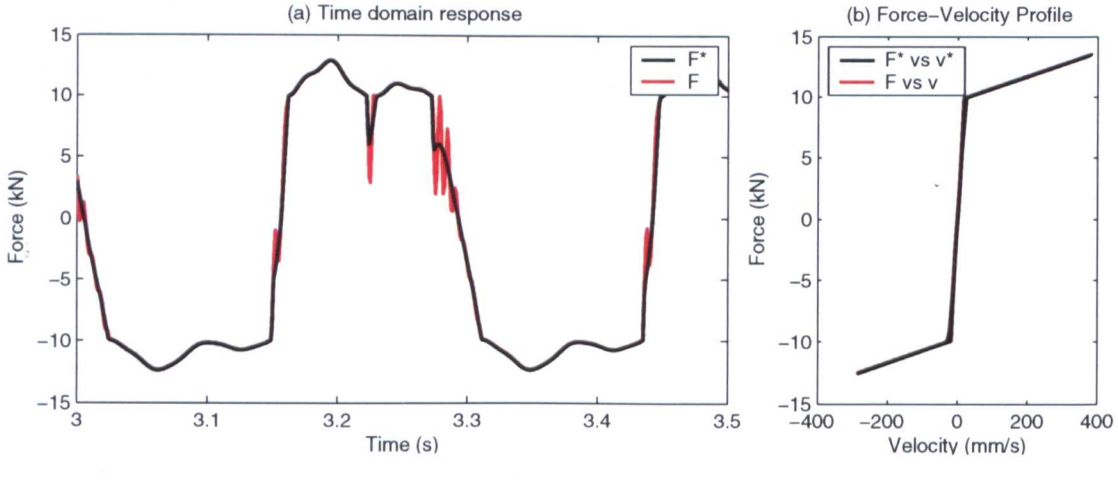


Figure 6.11: Repeat of test form Figure 6.10 with a discrete-time numerical substructuring algorithm ($\Delta t = 1$ms).

modelled relief valves open and close. It is important not to misinterpret this as an inaccuracy of how the substructuring algorithm has been setup, but instead to recognize that this is a limitation of the numerical modelling of the substructure. The output from the numerical model of the blade is a velocity that is fed directly into the numerical superstructure, which in turn outputs the relevant force purely on an explicit algorithm. This allows the force to dramatically change (unrealistically) around the critical region of the relief valve motion, as discussed in Eyres et al. [117]. When performing experimental substructuring however, the physical substructure is continuous (one of the main reasons for the developments of the technique) and thus will give a response similar to that of the continuous-time numerical substructuring algorithm, except containing the real, nonlinear dynamics of the damper.

6.8 Stability of the substructuring algorithm

Figure 6.12 shows a simplified schematic representation of the blade and lag damper emulated system decoupled for each mode $i = 1, \ldots, 8$. The nonlinear damper a_{4i} and the nonlinear spring a_{5i} are taken to be the substructure. For the real damper the level of damping (a_4) is dependent on the nonlinear hysteretic behaviour observed in the lag damper system identification of § 6.4.1. The nonlinear spring (a_5) represents the compressibility of the fluid inside the damper chambers. This

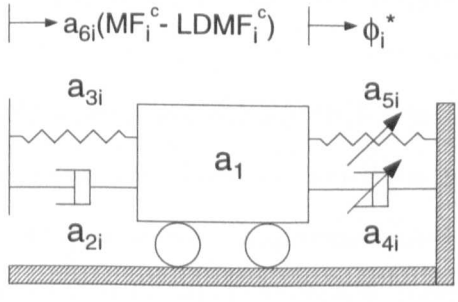


Figure 6.12: Schematic representation of the blade and lag damper system for each mode $i = 1, \ldots, 8$.

is a complex problem as the faster the lag damper's piston is moved the higher the effects of compressibility and the more the damper starts to behave like a spring rather than solely providing inertial damping. We can rewrite Eq. (6.2) in this simplified structure for the substructured system (analogous to the single-DOF case study Eq. (3.4)), such that

$$a_{1i}\ddot{\phi} + a_{2i}\dot{\phi} + a_{3i}\phi + a_{4i}\dot{x}_i + a_{5i}x_i = a_{6i}(MF_i^{code} - LDMF_i^{code}), \tag{6.23}$$

where, $a_{1i,\ldots,6i}$ are predetermined coefficients (calculated from the parameters defined in Eq. (6.2)) for each mode $i = 1, \ldots, 8$ (this data is commercially sensitive and therefore cannot be published), and again the state of the transfer system x_i is described by a unit delayed response of the numerical model ϕ_i, such that $x_i = \phi_i(t - \tau)$. Solving this DDE will create eight separate critical limits, $\tau_{c1,\ldots,8}$.

It is beyond the scope of this pilot study to perform a stability analysis on the real lag damper substructured system due to the highly nonlinear characteristics of the damper and the fact that the modes are decoupled in this way. Investigative work needs to be undertaken to see if passing one critical limit causes local or global instability of the substructuring algorithm, see future work § 7.2. However, as there is only one transfer system in this case, the smallest value determines the absolute stability of the overall substructuring algorithm.

To achieve this we perform an approximate stability analysis using the idealized viscous damper. Thus, compressibility will be ignored, $a_5 = 0$. The damping coefficient of the idealized viscous damper can be calculated by simplifying the damper characteristics to being approximately linear (rather than nonlinear) piecewise smooth and reading off the resultant gradients. This will produce two coefficients — c_1 for when the blow-off valves are both closed and c_2 for when one is open. As we have seen in Chapter 3, the critical limit of stability is based on a ratio between the damping and the stiffness of the substructure system. c_1 represents the case where the idealized damper has both high damping and high stiffness, whereas c_2 is the case for low damping and low stiffness. We therefore must consider both situations to see which

is the dominant case in terms of stability. We can rewrite Eq. (6.23) to give

$$a_{1i}\ddot{\phi} + a_{2i}\dot{\phi} + a_{4i}c_{1,2}\dot{\phi}(t-\tau) + a_{3i}\phi = 0, \tag{6.24}$$

for the unexcited system, as in the DDE analyses performed in Chapter 3.

We can now study the substructured system of Eq. (6.24) in order to determine the sixteen (eight for each damping case) critical delays τ_{ci} above which the system is unstable. As Eq. (6.24) is second-order DDE (as for the multi-DOF case study, § 3.2.5) we use DDE-BIFTOOL ([91]) to find real part of the characteristic root and therefore the absolute critical delay τ_{ca} (the smallest critical value and thus the delay magnitude which determines the absolute stability).

Figure 6.13 shows the real part of the characteristic roots for Eq. (6.24) for the damping case of c_1 where both blow-off valves closed. The dominant mode (the first root to cross the zero axis) is highlighted in bold and in fact represents the 7^{th} mode which models the third *lag* mode. The smallest critical value is shown in the enlarged view in Figure 6.13(b) and has a value of $\tau_{c7} = 0.75$ms. The smallest critical value for the damping case of c_2 is calculated to be $\tau_{c8} = 6.34$ms and represents the

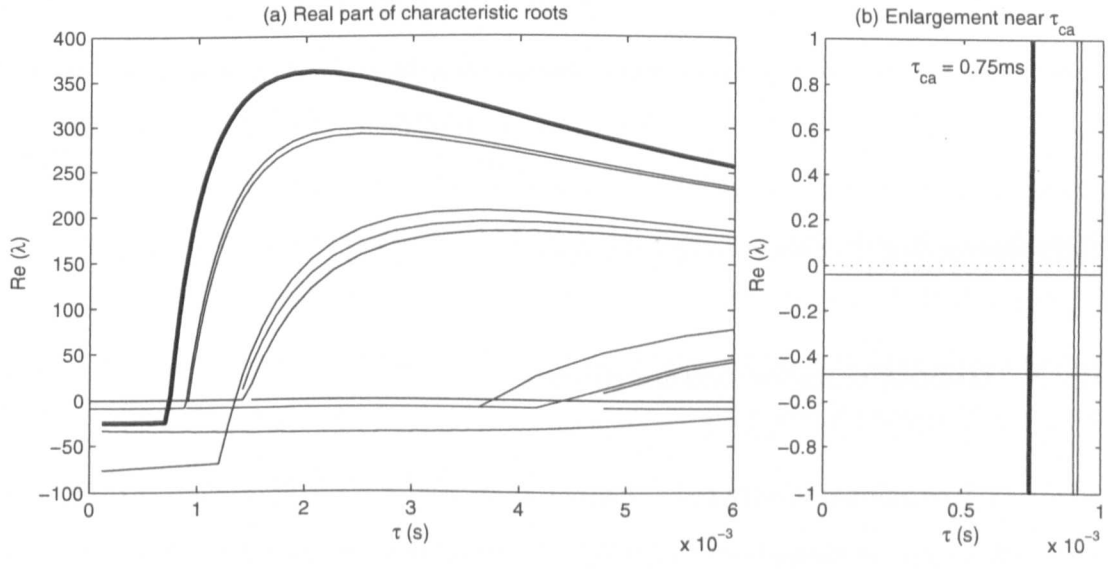


Figure 6.13: Real part of the characteristic roots for Eq. (6.24) for the damping case of c_1 (both blow-off valves closed).

8^{th} mode which models *twist*. Thus, the case where the blow-off valves are closed is shown to be the dominant factor in terms of stability, and sets the absolute critical value to be $\tau_{ca} = 0.75$ms.

However, we know this to be an approximation as the compressibility of the fluid has been ignored and only the simplified linearized system has been considered. Despite this, it does provide an initial guide to work from when performing an experimental test. It demonstrates the very high level of performance which is required due to the very small stability margin and thus why robustness is an essential consideration in terms of achieving a successful real-time dynamic substructuring test for an industrial application.

6.9 Experimental real-time substructure testing of a real lag damper

We take the general case of steady state flight at 84knots, as was used for the numerical substructuring algorithm of § 6.7. We use this criterion, along with a number of specific helicopter properties to find the constants for Eq. (6.2) — these then set our steady state flight conditions (the details of this information is industrially sensitive and therefore cannot be published). In order to gain the best chance of achieving a successful real-time dynamic substructure test we now apply the *robust transfer system design* of § 5.2.

6.9.1 Robust transfer system design

Step 1: Proprietary control. Rather than perform an open-loop system identification, as carried out in § 2.3.4, the Instron 8800 control hardware contains a self-tuning algorithm. This was performed for a PID controller and then fine tuned to give a damping of ≈ 0.8; the controller gains are shown in Table 6.2.

Gain	*Value* (dB)
P	32.0
I	1.2
D	0.8

Table 6.2: Fine tuned PID gains for EH101 test rig.

Step 2: Transfer system identification. The resultant characteristic performance of the actuator is good and highly repeatable with only low nonlinearity; there is still a dead zone which must be overcome during change of direction due to the static friction of the piston. However, it is noticeably better than the servo mechanical actuators used for the case study experiments.

A closed-loop transfer system identification was then carried out under a sine sweep excitation (from 0-10Hz in 60s at ±5mm) to produce the resultant nominal and uncertainty models.

Step 3: Nominal model. As in § 5.2 we use the MATLAB Output Error (*oe*) function to approximate the "best" fit of experimental response x compared to the sine sweep demand r. A 1^{st} order fit is shown to capture all the nominal characteristics and is given by

$$\frac{x(t)}{r(t)} = \frac{166.5}{\mathrm{s} + 169.3}. \tag{6.25}$$

As Eq. (6.25) is a simple 1^{st} order model it is straightforward to use as the gross feed-forward *cancellation* controller according to Eq. (5.3) described in § 5.2. The displacement and velocity states directly output from the numerical model of the bade can be explicitly used so the problem of having an improper transfer function is removed.

We can now observe the potential accuracy of this cancellation controller (i.e.: the ability of the nominal model to faithfully capture the dynamic behaviour of the transfer system) by repeating the numerical substructuring test of Figure 6.10 with the physical substructure running in parallel — this is in

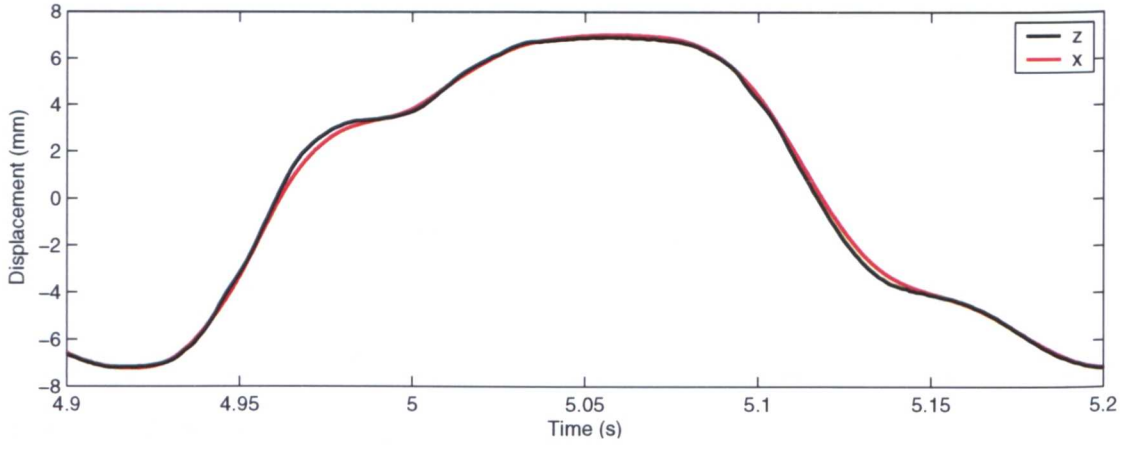


Figure 6.14: Accuracy of the *nominal* model cancellation controller for a substructuring test at a flight speed of 84knots (repeat of Figure 6.10); Numerical substructuring response: z, Experimental transfer system response: x.

fact an experimental real-time dynamic substructuring test with robustness compensation of $\gamma = 0$ (§ 5.3.3). Figure 6.14 shows the accuracy of the nominal model cancellation by comparing the *desired* numerical model displacement, z, of the numerical algorithm to the *actual* transfer system displacement, x, as an outer-loop controller.

Instead of using the nominal model the AFP algorithm could also be used to achieve the required level of delay compensation. However, the complexity of the substructured system is high due to the large discontinuity in the damping characteristics (at the instant a relief valve opens) resulting in subsequent discontinuities in the numerical model displacement, z, which would be difficult to compensate for using polynomial extrapolation — the AFP algorithm is strictly designed to achieve compensation of smooth signals. As the nominal model is so good in this case it will be used instead of the AFP algorithm as it is inherently more stable to non-smooth signals.

Step 4: Uncertainty model. We cannot compute an explicit model for the unmodelled dynamics as they are expressly nonlinear, but instead must be aware of when they are significant and how best to achieve a successful test. The

formalization of the stability criterion in § 6.8 (although approximated) and the good nominal model of the transfer system dynamics reduces the uncertainty in the substructuring algorithm substantially.

As discussed in Chapter 2, substructuring started in the field of large structural testing, specifically structures under the influence of an earthquake. In this situation, all transient behaviour of the structure is of vital importance for a good understanding of the full dynamic response. Therefore, the substructuring algorithm must utilise the experimental force for the entire duration of the test, a robust technique for achieving this is discussed in § 5.5. However, in the case of the EH101 lag damper, it is only the steady state dynamics that we are interested in and therefore it is unnecessary, and in fact meaningless, to start the test using the experimental force. This is due to that fact that it is assumed that the blade is forced periodically by a constant matrix in a constant state of trim condition. The entire test would only have to be operated using the experimental force if the numerical model of the blade was excited by flight data that started from rotor start up to helicopter take off before reaching steady state flight — this is beyond the scope of this pilot study, see § 7.2.

Thus, the most appropriate robustness compensation scheme to use is the γ-compensator as this has already been show in Step 3 to produce a stable substructuring algorithm (although γ was set to zero). Therefore, the experimental test can be started and run with $\gamma = 0$ while there is any transient behaviour and while the cancellation controller achieves steady-state synchronization. Then, using a linear progression of $\gamma = 0 \rightarrow 1$ in 5s (such that no high frequency modes are excited and any induced erroneous dynamics are removed) can be performed to achieve a fully experimental real-time substructuring test. Should the margin to instability not be sufficient then γ can be stopped before it reaches one such that only part of the experimental force is utilized.

The framework of robust transfer system design is flexible and allows the most appropriate algorithms to be selected and used in accordance to their need for a specific application. It is envisaged that the robustness compensation (Step 4) is only used when it is required as an inherent drawback is that robustness is always achieved in compromise of the dynamical accuracy of the numerical model.

6.9.2 Steady-state flight

We perform a real-time experimental substructuring test for the same excitation conditions as for the numerical substructuring test of Figure 6.10. The robust transfer system design is applied as stated in § 6.9.1 such that the test is commenced with $\gamma = 0$ to ensure stability. Typical experimental results are shown in Figure 6.15 for one continuous test. Figure 6.15(a1) shows the test between 6.6-7.6s for $\gamma = 0$

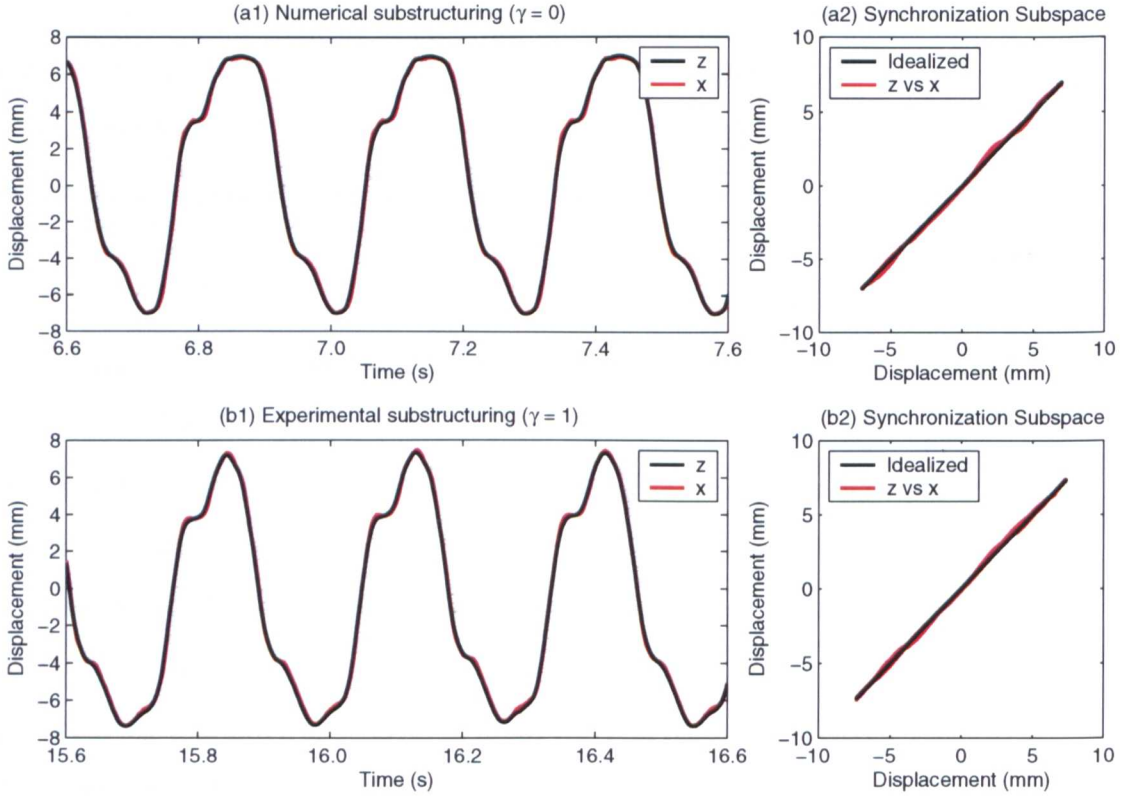


Figure 6.15: An experimental substructuring test at a flight speed of 84knots.

but after all transient behaviour has died away and the cancellation controller has achieved full delay compensation as can be seen from the synchronization subspace plot of Figure 6.15(a2). The robustness compensation is then removed over a 5s period to give Figure 6.15(b1) which shows the test between 15.6-16.6s for the situation of $\gamma = 1$ — this is now real-time dynamic substructuring test using 100% of the experimental force. The algorithm is stable due to the high level of synchronization which is still achieved by the nominal model inversion as can be seen in Figure 6.15(b2).

It can be seen that the characteristic shape of the steady-state numerical model displacement is different for the numerical case ($\gamma = 0$) to the experimental case ($\gamma = 1$) as expected. This is because the force signal fed back form the substructure is now representative of the true dynamics of the *real* lag damper (as shown in the lag damper system identification in § 6.4.1) rather than the dynamics of the idealized viscous damper (Figure 6.4). This can clearly be seen in Figure 6.16 showing the test at the same 15.6-16.6s interval when $\gamma = 1$. The numerical force signal is shown in black whereas the actual force signal being fed back from the substructure is shown in red as can be seen from Figure 6.16(b). The numerical force is being calculated

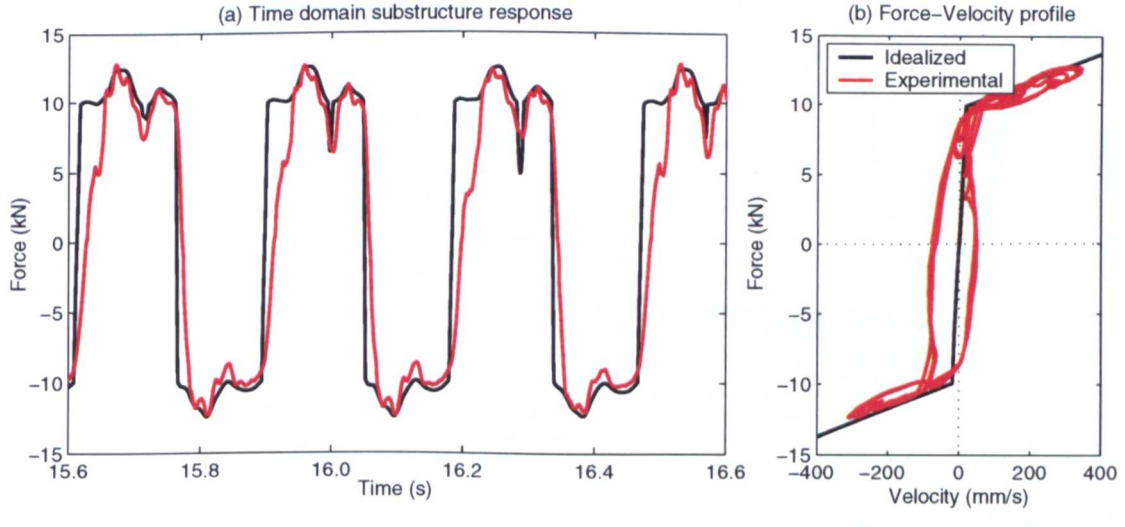


Figure 6.16: Force feedback during substructuring test from Figure 6.15.

in parallel to the experimental force being measured and shows how the idealized lag damper would behave at any given moment in time — in this case we know this is not actually representative of the true system.

As the system is so complex we have actually no way of calculating the emulated system. Therefore, we use the method presented in § 4.3.7 of observing the local control error, e_2, in conjunction with the capacity utilization of the actuator. The local error is clearly very small as can be seen from Figure 6.15(b2). Figure 6.17 shows the capacity utilization of the actuator against an estimated performance envelope for the actuator (based on a generic hydraulic actuator) for 5s of experimental data. It can be seen that the majority of the test is performed well below 50% of the actuator's capacity along with the whole profile being located well within the linear region. Thus, we can have high confidence that the global error for the experimental substructuring test is small and therefore this is demonstrative of the lag damper's true dynamic characteristic in situ during flight.

It is clear from Figure 6.16(a) how the characteristic hysteretic behaviour of the real damper manifests itself in altering the idealized response. Therefore, the red line

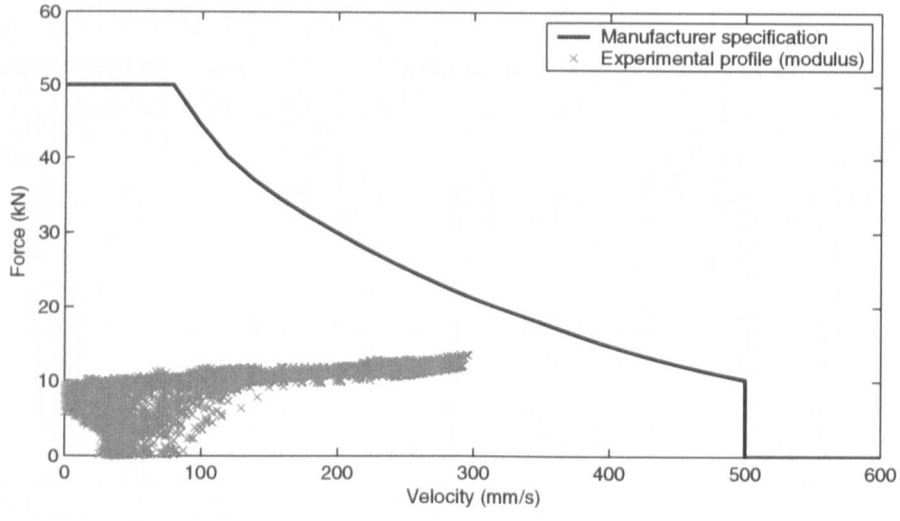


Figure 6.17: Estimated actuator capacity envelope for the actuator. Experimental data shown for the test of Figure 6.16 (5s test data).

(the experimental force signal) can provide us with a far greater understanding of the vibrational characteristics of the energy being transmitted back into the helicopter fuselage than the idealized model. This is because it contains the same modal frequency content as would be found from the same lag damper on an actual helicopter in steady-state flight at 84knots. This information can then be used to alter the characteristic dynamics of the lag damper by changing the tunable parameters (such as orifice size, bypass diameter, viscosity, relief valve arrangement and critical values etc.) to reduce in vibration transfer at the dominant modes of $n \pm 1$ the number of blades — however, this is beyond the scope of this work, see § 7.2.

6.9.3 Substructuring instability

As witnessed for the small scale case studies in Chapter 4, when the level of delay compensation is reduced such that the magnitude of the response delay τ is greater than the critical delay τ_c, instability is observed characterized by the onset of oscillation with positive exponential growth. However, in this case the absolute

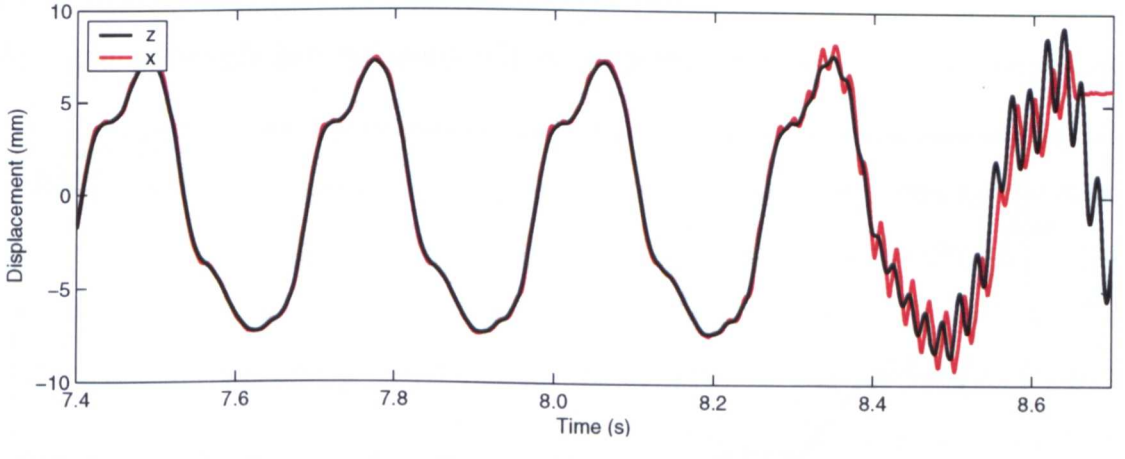


Figure 6.18: Progression to instability as the magnitude of delay compensation is reduced (after approximately 8.6s the failsafe system kicks into action and stops the test automatically).

critical limit, τ_{ca} (calculated in § 6.8) is only estimated. Figure 6.18 shows the case when the cancellation due to the nominal model is reduced. By increasing the value of the numerator and denominator (keeping the ratio the same to maintain the level of steady-state amplitude correction) the level of delay compensation is decreased until instability is observed at an approximate value of:

$$G_n'(s)^{-1} = \frac{\mathrm{s} + 245}{241}, \tag{6.26}$$

where $G_n'(s)$ is a reduced accuracy nominal model. At the dominant excitation frequency of 3.5Hz this corresponds to 1.8ms difference in the magnitude of delay compensation gained from using the original nominal model $G_n(s)$ of Eq. (6.25). Therefore, as we know the original nominal model provides a very high level of synchronization this gives an approximate experimental critical limit of $\tau_c \approx 1.8$mm rather than the approximated value of $\tau_{ca} = 0.75$ms. The reason for the increased margin of stability can be understood for two reasons. Firstly, the higher order terms of the nonlinear piecewise smooth function are discarded in the linear approximation and secondly, by referring to the experimental force-velocity profile of Figure 6.16(b). During the period when both blow-off valves are closed, the characteristic profile only attains the steepest gradient (as used by the linearized approximation for c_1) for a very short period of time before the corresponding blow-off valve opens. As discussed in Chapter 3, instability is not catastrophic, the substructuring algorithm must stay

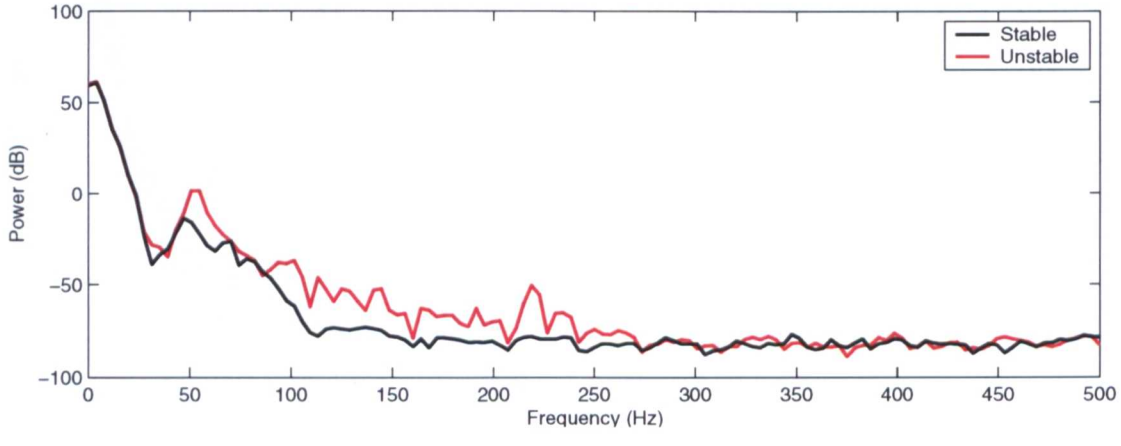


Figure 6.19: Spectrum of instability frequencies from Figure 6.18.

in the unstable region for cumulative time steps for the oscillations to build up (the rate is determined by the magnitude of the real part of the characteristic root at the given delay). It is therefore possible that the substructuring algorithm does pass into the unstable region before $\tau = 1.8$mm but does not lead to a catastrophic failure, as can be seen from the small oscillations building on the first rising peak. Figure 6.19 shows the frequency content of a stable substructuring algorithm compared to that of the unstable test of Figure 6.18. As there are now eight modelled modes in the numerical model along with the nonlinear characteristics of the damper fluid, instability is now observed over a whole range of frequencies.

6.9.4 A comparison of two different lag dampers

It is clear that the lag damper has a significant influence on the blade dynamics and thus the vibrational energy transferred back into the fuselage. Therefore, as the EH101 is a five bladed helicopter all the lag dampers must be balanced such that no erroneous dynamics are created in the hub.

Figure 6.20 shows a repeat of the test shown in Figure 6.15 for the constant flight speed of 84knots but for two individual lag dampers. Although the EH101 lag dampers are manufactured to strict tolerances they are only specified by the maxima/minima values of the testing points set shown in the system identification of the lag damper in § 6.4.1. No information is given about its specific dynamic profile. Figure 6.15 shows that although the dynamic characteristics of the two dampers are similar, their exact behaviour is not, even though they are excited by the same flight data under the same testing conditions. Here, this is due to the fact that the second damper is no longer flight certified, but this comparative test does show how a set of lag dampers could be balanced for an individual hub system and thus reduce vibration transfer to the rest of the helicopter.

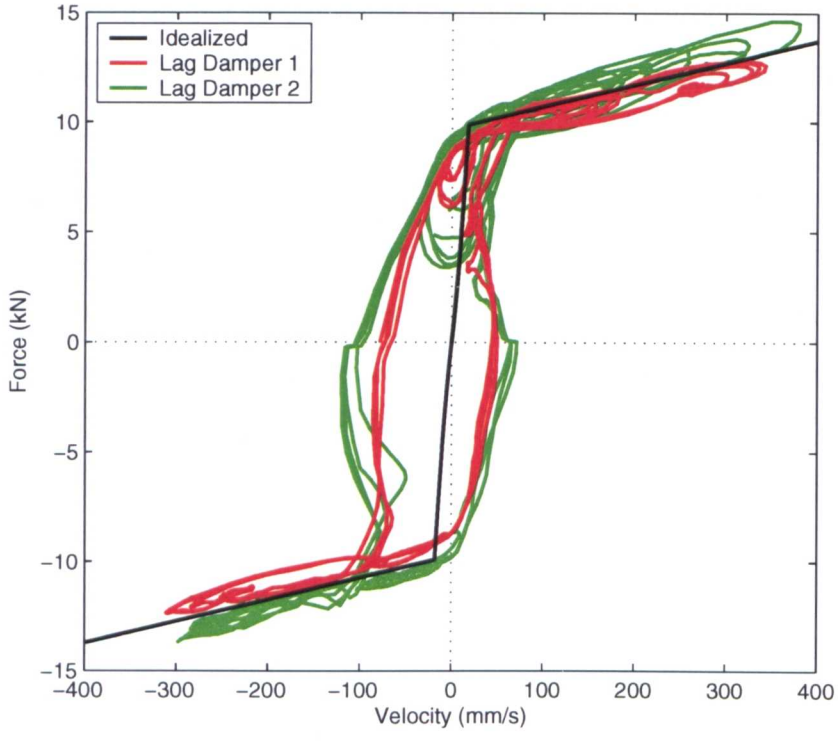


Figure 6.20: A comparison of two different lag dampers for an experimental substructuring test (a repeat of the test from Figure 6.15).

6.10 Conclusion

Helicopters vibrate. An unadjusted helicopter can easily vibrate so much that structural damage can be caused or create an operational environment for the pilot that can only be sustained for short periods of time. To reduce vibration, all fully articulated helicopters have a combination of three hinges. Because the advancing blade has higher airspeed than the retreating blade, a perfectly rigid blade would generate more lift on that side and tip the aircraft over. In consequence, rotor blades are designed to "flap" — *lift* and *twist* in such a way that the advancing blade flaps up and develops a smaller angle of attack, thus producing less lift than a rigid blade would. Conversely, the retreating blade flaps down, develops a higher angle of attack, and generates more lift. At high speeds, the force on the rotors is such that they flap excessively and the retreating blade can reach too high an angle and stall unless properly damped. The lag damper is the most efficient current means

for controlling this motion, however, its inclusion in this dynamic system introduces its own problems.

We have shown here how real-time dynamic substructuring can be used to better understand the effect of the lag damper on the entire system rather than observing its dynamic characteristics in isolation, as in the end, the actual force-velocity characteristic of an individual lag damper is immaterial, the important criterion is its dynamic effect on the entire helicopter.

The robust transfer system design has been used here to good effect in order to achieve a range of successful experimental tests (§ 6.9.1). The stability of the substructured system was studied in § 6.8, and although approximated gave a good working indication for the performance criteria of the delay compensation scheme. In this case we apply an inverse of the nominal model (for Step 3) to act as a gross feed-forward cancellation device rather than the adaptive AFP algorithm presented in Chapter 4. This was done for two reasons; first, the excitation is constant and periodic such that the need to adapt to changing plant dynamics during experimentation is not required, and secondly, the large discontinuity in the force characteristic (due to the piecewise smooth nature of the lag damper) introduces corresponding discontinuities in the numerical model displacement, the demand signal for the transfer system. As stated earlier, the AFP algorithm is strictly designed to achieve compensation of smooth signals and thus achieving accurate compensation of this particular numerical model signal using polynomial extrapolation is difficult. However, as the framework of robust transfer system design is flexible this allows the most appropriate algorithms to be selected and used in accordance to their need as shown by the γ-compensator only being used at the start of the test (Step 4). Once steady state conditions were achieved all robustness compensation was removed resulting in the complete experimental dynamics from the substructure to be fed back into the substructuring algorithm.

These results show a validation for this testing technique for smaller scale mechanical

component testing and thus a broadening of the original concept of application of large scale structural seismic testing.

Chapter 7

Conclusions and recommendations

SUMMARY: This chapter provides a summary of the work detailed throughout this thesis. We highlight the main advances made by the work and make recommendations for potential future directions.

7.1 Discussion

The basis of the work presented in this thesis is on understanding the fundamental principles behind the experimental side of the hybrid numerical-experimental testing technique known as *real-time dynamic substructuring*. The complexity of the substructuring technique arises due to "gluing" the numerical and experimental parts of the virtual testing environment together. The influence of the numerical model(s) must be applied to experimental specimen(s) through physical device(s), usually an actuator, but as with any piece of equipment the *transfer system* (as it is known) has its own dynamics which must be included within the substructured system. The typical consequence is the introduction of *delay errors* into the calculation of the numerical model(s). Due to the bi-directional coupling between the experimental and numerical side of the virtual environment this error leads first to a loss of dynamical accuracy of the substructured system (compared to the complete structure) before instability is observed, characterized by the onset of oscillations

with exponential growth. This marks a failed test and can result in catastrophic damage being sustained by the substructure.

At the time the research for this thesis was started, *real-time* dynamic substructuring had just moved beyond the "proof of concept" stage. Now, the field is maturing into a viable testing platform. The main contribution of this thesis has been to develop techniques in order to overcome the phenomena of delay induced instability and apply them to a mechanical and aerospace industrial application. The following is a summary of the main areas that are covered in this thesis and the advances that have been achieved:

Formalising a generic substructuring algorithm

This is discussed in detail in Chapter 2 and demonstrated by Wallace et al. [69]; work in this area has also been published in [70, 118, 119]. The generic substructuring control algorithm was split up into two separate feedback loops. The *inner-loop* which contains the actuator and its proprietary (linear) controller in order to form the *transfer system* (whose primary purpose is to achieve a suitable "linear" response such that the effects of uncertainty and nonlinearity are reduced to an acceptable level) and the *outer-loop* which controls the level of delay compensation and magnitude of the robustness compensator. The outer-loop can either be a fixed feed-forward process or an adaptive feed-back algorithm using the same synchronisation error, e_2, as the inner-loop control. This error is also the only measurable quantity of accuracy when the substructured system is complex as discussed later. Through the use of concepts taken from synchronization theory we are able to observe this error online during the experimental testing which has the advantages of:

- giving the user an initial and instant guide to the accuracy of an individual test and where the potential sources of errors may lie instead of carrying out lengthy post processing analysis.

- allowing linear controllers to be tuned online without the need for system identification.
- viewing the adaption characteristics of adaptive controllers.

These can be extremely useful in gaining a greater understanding of the experimental characteristics and limitations of the experimental equipment along with highlighting the major contributing factors that limit the level of synchronisation.

Finally, by treating the feedback force from the substructure (in the outer-loop of the substructuring algorithm) as an autonomous "disturbance" both the numerical models and control of the transfer systems become decoupled for a multi-DOF or multi-transfer system substructured system. In the case of the case study examples, this allowed the equations of motion of the numerical model to be solved as an ODE, disregarding the inherent delay in the substructuring algorithm, thus allowing fast computation essential for real-time control.

As long as the strict constraints of hard real-time control are adhered to, any type of numerical modelling technique and any type of control algorithm can be utilized. Thus, there is great flexibility in this generic framework for speciallizing the specific algorithms to suit the specific substructured system.

A formal stability study on the effect of delay errors

A formal stability analysis of delay induced instability was performed in Chapter 3 and can also be found in Wallace et al. [69] and by Gawthrop et al. [74]; work in this area has also been published in [120–122]. The transfer system(s) which act on the substructure are controlled to follow the appropriate output from each respective numerical model, which is typically a displacement. Thus, delays arise naturally as it is not possible for any controlled system to react instantaneously to a change of state as prescribed by the numerical model. In fact, there are a number of different delays which combine together to give the overall delay of the

transfer system (including data acquisition, computation, digital signal processing and the actuator delay itself) which are combined into the overall synchronisation error, e_2. At the same time the force(s) between the transfer system(s) and the substructure are fed back into the numerical model(s) to give a form of bi-directional coupling which is affected accordingly by the same error and thus is the source of the instability. It is important to note that this form of instability is different to the numerical instability encountered from an integration algorithm.

We studied the stability of the substructured system in direct relation to the magnitude of this delay error and presented two methods for identifying the critical limit of stability, the maximum value the delay of the transfer system can reach before instability was observed, characterized by the onset of exponentially growing oscillations. The method of modelling the substructured system with delay differential equations (DDEs) explicitly includes the delay(s) due to the transfer system(s) [69]. The advantage of DDE modelling is that powerful analytical and numerical methods can be used to determine the stability of the DDE model and, hence, of the substructured system. The phase margin approach [74] provides a measure of how near to instability the ideal system is in terms of how much phase lag is permissible. The disadvantage of this approach is that the substructured system must be approximated to a linear transfer function. However, it can be readily used for the class of systems for which the DDE methods cannot or when it is impractical to use other techniques, giving it considerable advantages in terms of practical implementation.

We have shown explicit calculations for the stability limits using both approaches before demonstrating a numerical approach which has the advantage of being suitable for complex and nonlinear systems with more than one delay. This remains a source of future work as discussed in § 7.2.

The application of delay compensation to real-time substructuring

Due to the inherent delay introduced into the substructuring algorithm, as shown by the delay induced instability, delay compensation techniques are required in order to achieve stability and are discussed in Chapter 4 and by Wallace et al. [75] and Gawthrop et al. [76]; work in this area has also been published in [120]. These delay compensation techniques are different in principle to the classical control of a delayed system as they are inevitably linked to the stability of the overall algorithm through the magnitude of the delay which remains.

We discussed two different formulations of a compensation scheme to be used as an outer-loop controller. The first type assumes that the dynamics of the transfer system may be approximated to a pure delay and utilizes forward prediction methods using polynomial extrapolation to predict forward the numerical model displacement. This was achieved through calculating the polynomial coefficients each time step such that non-integer multiples of a time step can be predicted and also incorporates an amplitude correction algorithm [75]. This method was extended by vary the amount of delay compensation in an adaptive manner based on the error between the actuator displacement and the desired numerical model displacement, e_2. This, in effect, relaxes the assumption of a pure delay and thus can account for variable plant dynamics. The second type was achieved via lag compensation [76]. By estimating an experimental transfer function of the combined inner-loop controller and actuator dynamics the outer-loop controller can compensate for unwanted dynamics by applying the inverse of the transfer function estimation. This has the advantage of achieving compensation over the entire frequency range of operation for the transfer system.

The concept of achieving a robust substructuring algorithm

The work described in Chapter 5 and by Gawthrop et al. [74] extends the outer-loop control strategy in order to introduce the concept of robustness. Robustness is an

essential consideration in the formulation of a successful testing strategy as it reduces the uncertainty of the transfer system response and increases the available margin to the critical limit of stability. However, this was always achieved in compromise of the dynamical accuracy of the numerical model. This work led us to develop a four stage testing methodology that can be applied to any substructured system to help ensure successful testing.

How accuracy can be assessed for complex substructured systems

The question of accuracy in terms of a substructuring algorithm is complex and discussed throughout the thesis and by Wallace et al. [75]; work in this area has also been published in [70, 118]. The problem is complex as there are two coupled errors, the *numerical model* error, e_1, and the *local* control error, e_2, which combine together as described in Proposition 1 (§ 2.6) such that as the local error is reduced, the numerical model error is also reduced ($z \rightarrow z^*$ if $x \rightarrow z$); although, the exact relationship between e_1 and the e_2 will be system dependant. These combine to give the overall accuracy of the substructuring experiment comparing the dynamics of the *virtual* substructuring test to that of a test for the *complete* structure (should that be possible) and is called the *global* error.

However, it is the typical situation in substructuring that the emulated dynamics are not known and is in fact why the substructuring tests are being performed. Therefore, the numerical model error, e_1, and thus the global error could never be explicitly calculated. We described in Chapter 4 ([75]) a measure of the accuracy for a substructuring test without having to simulate the complete system and use this approach in Chapter 6 where the emulated dynamics are unknown. This was achieved by observing the local control error, e_2, in conjunction with the capacity utilization of the actuator compared to its manufacture's linear performance envelope. Although this method only provides a confidence value to the experiential substructuring test, it is a generic result which holds for all substructured systems.

An industrial example of real-time dynamic substructuring

The substructuring experiment discussed in Chapter 6 represents the current "state of the art" in real-time dynamic substructuring as it is one of the first tests of its kind at the time of print. Other substructuring experiments have being performed on industrial sized dampers, such as Horiuchi et al. [61], however none which involve such complex dynamics of both the substructure and the numerical model for an active commercial project. We build on all the fundamental concepts introduced and developed through Chapter 1 to Chapter 5 in order to achieve successful substructure testing of the AgustaWestland EH101 lag damper connected to a numerical model of an individual blade excited by flight test data. The lag damper itself is highly complex due to its nonlinear piecewise smooth hysteretic characteristics and due to its situation highly impractical to monitor during flight. Additionally, due to flight regulations/specifications a modified lag damper cannot be used on an active helicopter and thus means that testing of the lag damper's influence on the entire helicopter is extremely difficult. Any other current experimental testing technique would either loose the true dynamic behaviour of the lag damper or observe its dynamic characteristic only in isolation.

We have clearly demonstrated the "proof of concept" of real-time dynamic substructuring for such a complicated mechanical substructured system and thus a broadening of the real-time technique from seismic testing of large civil engineering structures. This leads us to be able to develop the substructuring experiment into an increasingly involved test of a complete helicopter in variable flying conditions as discussed in future work, § 7.2.

7.1.1 General conclusions

The work presented in this thesis is a detailed analysis of the fundamental principles behind the experimental side of real-time dynamic substructuring technique. Build-

ing on the understanding gained we use this hybrid technique in order to achieve successful testing of the substructured EH101 helicopter lag damper. This is an example of an industrial scale problem where no other current experimental testing technique could reveal the level of information that is achieved through real-time dynamic substructuring.

7.2 Recommendations for future work

The following sections are suggestions of some potential lines of work that the author feels would complement and extend the work presented in this thesis:

Numerical modelling techniques

The work presented in this thesis is concerned with the development only of the experimental side of real-time dynamic substructure testing. However, the accuracy of the numerical modelling techniques used is an essential part of achieving a substructuring algorithm that can faithfully reproduce the dynamics of the original structure. A vein of current research is into utilizing finite element analysis (FEA) for the numerical model. However, at the current time, such techniques are limited by computer power as the entire model computation must exclusively take place within each time step Δt. Typically, real-time substructuring has a small sample time, no more than a few milliseconds, thus putting a high demand on the CPU. However, Moore's law[1] is a rule of thumb in the computer industry about the growth of computing power over time and states that the growth of computing power follows an empirical exponential law. Specifically, the total number of transistors on a CPU (and therefore the number of calculations possible) will double every 18 months (this is expected to hold at least until 2020). Thus higher computationally expensive numerical modelling techniques will become increasingly practicable in the following years to come.

[1] Attributed to Gordon E. Moore, the co-founder of Intel.

Additionally, the integration strategies which are employed by the real-time algorithm are also important, currently themes of research which started in PsD testing on mixed explicit-implicit algorithms are being applied for the first time in real-time dynamic substructuring by Wu et al. [80].

DDE analysis

Although a complete analytical and numerical DDE analysis was performed in Chapter 3, this was only for the conceptually simple substructured systems. The analytical power of the numerical analysis (DDE-BIFTOOL) can be applied to far more complex systems with many degrees of freedom and many different variable delays. First, an analysis should be done with the multi-DOF substructured system (§ 2.3.2) for two independent delays, τ_1 and τ_2, for each of the independent transfer systems. This will provide an insight into the coupling of the individual delay terms on as simple a system as possible. This will result in an understanding of local and global stability — whether passing one limit will always lead to instability.

Additionally, a complete numerical analysis for the EH101 lag damper experiment could be performed including the compressibility of the oil within the damper itself, details of which are given in Eyres et al. [117]. This is a complex problem as the faster the lag damper's piston is moved the higher the effects of compressibility and the more the damper starts to behave like a spring rather than solely providing inertial damping. This will have a serious effect on the critical limit of stability as the stiffer the structure the smaller the margin to instability.

Robust adaptive delay compensation

The AFP algorithm of Chapter 4 was shown to achieve high levels of synchronisation for variable transfer system dynamics. However, there are a number of issues associated with its use:

- Polynomial extrapolation is fundamentally unstable.
- Adaption is only achieved on set state conditions.
- The piece-wise smooth nature of the EH101 damper causes problems due to the non-smooth nature of the numerical model demand produced when excited by the flight test data.

This leads us to the need for the development of a more robust delay compensation algorithm which is already a current line of research in real-time testing. The most comprehensive of strategies would model the transfer system dynamics on-line such that any change in behaviour or nonlinear characteristics could be cancelled out. However, whatever this new algorithm it will still be applicable to fit into the framework of the robust transfer system design methodology of Chapter 5.

The EH101 lag damper project

The work presented in this thesis is a full validation of the real-time substructuring technique for this application, however, it does open the possibility for far more involved substructured systems to be analysed. The following is a summary of the potential lines of research in terms of the EH101 lag damper project, each one building on the ground laid by the previous one:

Numerical model validation A validation of the parametric lag damper model by Eyres [2] is required. If its accuracy and limitations can be understood by comparing the results to similar experimental substructuring data, it can then be used as a powerful analytical tool for initial design criteria when access to the full experimental substructuring rig is not available. Thus, we can have confidence in initial development without the need to ensure real-time performance.

Reduction in vibration transfer Typically, the dominant vibration transferred into the helicopter fuselage are of the modes $n \pm 1$ the number of blades.

Using substructuring, the tunable elements of the passive lag damper (such as orifice size, bypass diameter, viscosity, relief valve arrangement and critical values etc.) can then be analysed in isolation and the individual effects on the helicopter understood decoupled from the full complex dynamics.

Increased complexity of the blade numerical model Rather than substructure an individual blade and lag damper as we have done here, a more involved multi-bladed numerical model could be developed. The entire coupled five-bladed EH101 rotor system could then be modelled as an integrated unit, with five individual experimental lag dampers being tested at one time. Additionally, the analysis only uses 8 modes (4 flap modes, 3 lag modes and one twist mode). The numerical model could be extended to take into account the nonlinear characteristics of the composite blades for example.

Online varying of flight conditions All the work to date has concentrated on steady state flight assuming the blades are forced periodically by a constant matrix, modified to take into account the new force from the experimental lag damper. A long term goal would be to encompass flight dynamics from an entire flight, from take off to landing and including transient flying conditions such that the modal matrices are calculated on-line.

Smart damping techniques Recently there has been a considerable research effort into smart damping devices. In order to reduce the energy absorbed by the lag damper (dissipated as heat), reduce specific vibration transfer and to adapt to differing environmental conditions smart damping devices change the amount of damping force generated adaptively during flight. However, these devices are still only in the experimental development stage. Substructuring would allow for far more demanding and rigorous testing of such dampers, speeding up the development process and increasing confidence in a full scale test on a real helicopter. In this case, the substructuring algorithm would simply run in the background, i.e. creating the virtual testing environment,

while the adaptive control of the *smart* parameters was tested. It is envisaged that a form of *higher harmonic control* will be used such as currently used on tilt rotor aircraft. This will allow active cancellation of in-plane shear loads during flight along with potentially removing induced vertical vibration.

General recommendations

The field of real-time dynamic substructuring is maturing due to the increased understanding of the fundamental concepts, some of which have been discussed in this thesis, and can now be viewed as an effective testing technique. At the University of Bristol there are current research projects into substructuring a complete car and motorbike suspension unit (on the small scale component testing front) and into substructuring of an individual cable section from a cable-stay bride (on the large civil structures side). However, more real-life applications need to be studied in detail as the technique is still viewed with scepticism by some in industry.

If we take the EH101 lag damper as an example, all of the future topics are interested in improving the effect of the lag damper on the overall helicopter and not just the dynamics of the individual damper. This is the foresight that real-time dynamic substructuring gives the designer/engineer; in the end, the force-velocity characteristic of an individual lag damper is immaterial, the important criteria is its dynamic effect on the entire helicopter. Substructuring is the most effective technique to date that allows an engineer to understand this complex relationship.

Appendices

Appendix A

Multi-DOF numerical model S-Function

SUMMARY: S-Functions are stand-alone C modules, written in a predetermined way that Simulink can understand and are implemented in a way that is directly analogous to that of model code. They contain their own public registration function (which is called by the top-level model code) that initializes static function pointers in its SimStruct. When the top-level model needs to execute the S-function, it does so via function pointers. The real-time model data structure encapsulates model data and associated information necessary to fully describe the model. This is an example S-Function for the numerical model of the multi-DOF case study of § 2.3.2.

```
/*  File    : multiDOF_NM.c
 *  Author  : Max Wallace
 *  Date    : 13/04/2004
 *  Abstract:
 *     Substructuring numerical model for the multi-DOF case study.
 */
```

```
#define S_FUNCTION_NAME multiDOF_NM              //  S-Function name
#define S_FUNCTION_LEVEL 2                       //  Class 2 S-Function

#include "simstruc.h"              //  Include public registration function

#define U(element) (*uPtrs[element])            //  Pointer to Input Port0

/*====================*
 * S-function methods *
 *====================*/

/* Function: mdlInitializeSizes ===============================================
 * Abstract:
 *    The sizes information is used by Simulink to determine the S-function
 *    block's characteristics (number of inputs, outputs, states, etc.).
 */
static void mdlInitializeSizes(SimStruct *S) {
    ssSetNumSFcnParams(S, 0);                    //  Number of expected parameters
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        return;                                  //  Parameter mismatch will be
    }                                            //  reported by Simulink

    ssSetNumContStates(S, 4);                    //  Number of continuous states
    ssSetNumDiscStates(S, 0);

    if (!ssSetNumInputPorts(S, 1)) return;       //  Number of input ports
    ssSetInputPortWidth(S, 0, 17);               //  Number of input variables
    ssSetInputPortDirectFeedThrough(S, 0, 0);    //  Set input usage requirements

    if (!ssSetNumOutputPorts(S, 1)) return;      //  Number of output ports
    ssSetOutputPortWidth(S, 0, 4);               //  Number of output variables

    ssSetNumSampleTimes(S, 1);                   //  Fixed single step sample time
    ssSetNumRWork(S, 0);
    ssSetNumIWork(S, 0);
```

```
    ssSetNumPWork(S, 0);
    ssSetNumModes(S, 0);
    ssSetNumNonsampledZCs(S, 0);

    /* Exception free code specified to increase performance */
    ssSetOptions(S, SS_OPTION_EXCEPTION_FREE_CODE);
}

/* Function: mdlInitializeSampleTimes =========================================
 * Abstract:
 *    Specifiy that we have a continuous sample time.
 */
static void mdlInitializeSampleTimes(SimStruct *S) {
    ssSetSampleTime(S, 0, CONTINUOUS_SAMPLE_TIME);      //  Inherit sample time
    ssSetOffsetTime(S, 0, 0.0);
}

#define MDL_INITIALIZE_CONDITIONS /* Function: mdlInitializeConditions ==========
 * Abstract:
 *    Initialise continuous state to zero.
 */
static void mdlInitializeConditions(SimStruct *S) {
    real_T *x0 = ssGetContStates(S);
    int_T  i;

    for (i = 0; i <= 3; i++) {
        *x0 = 0.0;                              // Set initial states to zero
    }
}

/* Function: mdlOutputs =======================================================
 * Abstract:
 *      Output states
 */
static void mdlOutputs(SimStruct *S, int_T tid) {
```

```
    real_T            *z    = ssGetOutputPortRealSignal(S,0);
    real_T            *x    = ssGetContStates(S);
    InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);

    /* output z */
    z[0] = x[0];                                    //  z_1
    z[1] = x[2];                                    //  \dot{z}_1
    z[2] = x[1];                                    //  z_2
    z[3] = x[3];                                    //  \dot{z}_2

}

#define MDL_DERIVATIVES /* Function: mdlDerivatives =============================
 * Abstract:
 *      Calculate derivatives
 */
static void mdlDerivatives(SimStruct *S) {
    real_T            *dx   = ssGetdX(S);
    real_T            *x    = ssGetContStates(S);
    InputRealPtrsType uPtrs = ssGetInputPortRealSignalPtrs(S,0);
    real_T            F1, rd1, rv1;
    real_T            F2, rd2, rv2;
    real_T            m_1, m_2, m_3;
    real_T            k_1, k_2, k_31, k_32;
    real_T            c_1, c_2, c_31, c_32;

    /* Numerical Model Inputs */
    rd1 = U(0);  rv1 = U(1);
    rd2 = U(2);  rv2 = U(3);
    F1 = U(4);   F2 = U(5);

    /* System Parameters */
    m_1 = U(6); m_2 = U(7); m_3 = U(8);
    k_1 = U(9); k_2 = U(10); k_31 = U(11); k_32 = U(12);
    c_1 = U(13); c_2 = U(14); c_31 = U(15); c_32 = U(16);
```

```
    /* States of Eq. (2.6) */
    dx[0] = x[2];
    dx[1] = x[3];
    dx[2] = (F1 / m_1) - ((c_1 / m_1)*(x[2] - rv1)) - ((k_1 / m_1)*(x[0] - rd1));
    dx[3] = (F2 / m_2) - ((c_2 / m_2)*(x[3] - rv2)) - ((k_2 / m_2)*(x[1] - rd2));

}

/* Function: mdlTerminate =====================================================
 * Abstract:
 *    No termination needed, but we are required to have this routine.
 */
static void mdlTerminate(SimStruct *S) {
}

#ifdef  MATLAB_MEX_FILE          //  Is this file being compiled as a MEX-file?
#include "simulink.c"            //  MEX-file interface mechanism
#else
#include "cg_sfun.h"             //  Code generation registration function
#endif
```

Appendix B

EH101 blade numerical model S-Function

SUMMARY: This S-Function works with a constants file ("init_const.m") which is picked up in the initialisation phase to set the global system parameters. Pointers to these parameters are defined at the start of the file. This information is commercially sensitive and therefore cannot be published.

```
/*  File    : EH101Blade_NM.c
 *  Author  : Max Wallace
 *  Date    : 12/09/2005
 *  Abstract:
 *      Substructuring numerical model for the Westland EH101 Lag Damper project.
 */

#define S_FUNCTION_NAME EH101Blade_NM   //  S-Function name
#define S_FUNCTION_LEVEL 2              //  Class 2 S-Function

#include "simstruc.h"                   //  Include public registration function
#include "math.h"                       //  Include math function for trigonometry
```

```
#define U(element)  (*uPtrs[element])   //  Pointer to Input Port0
#define U1(element) (*uPtrs1[element])  //  Pointer to Input Port1
#define U2(element) (*uPtrs2[element])  //  Pointer to Input Port2


#define PARAM_1(S) ssGetSFcnParam(S,0)  //  Pointers to global system parameters
#define PARAM_2(S) ssGetSFcnParam(S,1)
#define PARAM_3(S) ssGetSFcnParam(S,2)
#define PARAM_4(S) ssGetSFcnParam(S,3)
#define PARAM_5(S) ssGetSFcnParam(S,4)
#define PARAM_6(S) ssGetSFcnParam(S,5)
#define PARAM_7(S) ssGetSFcnParam(S,6)
#define PARAM_8(S) ssGetSFcnParam(S,7)
#define PARAM_9(S) ssGetSFcnParam(S,8)
#define PARAM_10(S) ssGetSFcnParam(S,9)
#define PARAM_11(S) ssGetSFcnParam(S,10)
#define PARAM_12(S) ssGetSFcnParam(S,11)
#define PARAM_13(S) ssGetSFcnParam(S,12)
#define PARAM_14(S) ssGetSFcnParam(S,13)
#define PARAM_15(S) ssGetSFcnParam(S,14)
#define PARAM_16(S) ssGetSFcnParam(S,15)
#define PARAM_17(S) ssGetSFcnParam(S,16)
#define PARAM_18(S) ssGetSFcnParam(S,17)
#define PARAM_19(S) ssGetSFcnParam(S,18)
#define PARAM_20(S) ssGetSFcnParam(S,19)
#define PARAM_21(S) ssGetSFcnParam(S,20)
#define PARAM_22(S) ssGetSFcnParam(S,21)
#define PARAM_23(S) ssGetSFcnParam(S,22)
#define PARAM_24(S) ssGetSFcnParam(S,23)
#define PARAM_25(S) ssGetSFcnParam(S,24)
#define PARAM_26(S) ssGetSFcnParam(S,25)
#define PARAM_27(S) ssGetSFcnParam(S,26)
#define PARAM_28(S) ssGetSFcnParam(S,27)
#define PARAM_29(S) ssGetSFcnParam(S,28)
#define PARAM_30(S) ssGetSFcnParam(S,29)
#define PARAM_31(S) ssGetSFcnParam(S,30)
```

```
#define PARAM_32(S) ssGetSFcnParam(S,31)
#define PARAM_33(S) ssGetSFcnParam(S,32)
#define PARAM_34(S) ssGetSFcnParam(S,33)
#define PARAM_35(S) ssGetSFcnParam(S,34)
#define PARAM_36(S) ssGetSFcnParam(S,35)
#define PARAM_37(S) ssGetSFcnParam(S,36)
#define PARAM_38(S) ssGetSFcnParam(S,37)

/*====================*
 * S-function methods *
 *====================*/

/* Function: mdlInitializeSizes ===============================================
 * Abstract:
 *    The sizes information is used by Simulink to determine the S-function
 *    block's characteristics (number of inputs, outputs, states, etc.).
 */
static void mdlInitializeSizes(SimStruct *S) {
    ssSetNumSFcnParams(S, 38);                  //  Number of expected parameters
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
        return;             //  Parameter mismatch will be reported by Simulink
    }

    ssSetNumContStates(S, 21);                  //  Number of continuous states
    ssSetNumDiscStates(S, 0);

    if (!ssSetNumInputPorts(S, 3)) return;      //  Number of input ports
    ssSetInputPortWidth(S, 0, 1);               //  Input variables for Port0
    ssSetInputPortDirectFeedThrough(S, 0, 1);   //  Usage requirements for Port0
    ssSetInputPortWidth(S, 1, 2);               //  Input variables for Port1
    ssSetInputPortDirectFeedThrough(S, 1, 1);   //  Usage requirements for Port1
    ssSetInputPortWidth(S, 2, 1);               //  Input variables for Port2
    ssSetInputPortDirectFeedThrough(S, 2, 1);   //  Usage requirements for Port2

    if (!ssSetNumOutputPorts(S, 1)) return;     //  Number of output ports
```

```
    ssSetOutputPortWidth(S, 0, 1);              //  Number of output variables

    ssSetNumSampleTimes(S, 1);                  //  Fixed single step sample time
    ssSetNumRWork(S, 8);                        //  Number of work functions
    ssSetNumIWork(S, 0);
    ssSetNumPWork(S, 0);
    ssSetNumModes(S, 0);
    ssSetNumNonsampledZCs(S, 0);

    /* Exception free code specified to increase performance */
    ssSetOptions(S, SS_OPTION_EXCEPTION_FREE_CODE);
}

/* Function: mdlInitializeSampleTimes =========================================
 * Abstract:
 *    Specifiy that we have a continuous sample time.
 */
static void mdlInitializeSampleTimes(SimStruct *S) {
    ssSetSampleTime(S, 0, CONTINUOUS_SAMPLE_TIME);      //  Inherit sample time
    ssSetOffsetTime(S, 0, 0.0);
}

#define MDL_INITIALIZE_CONDITIONS /* Function: mdlInitializeConditions ==========
 * Abstract:
 *    Initialise continuous states to zero.
 */
static void mdlInitializeConditions(SimStruct *S) {
    real_T *x0 = ssGetContStates(S);
    int_T  i;

    for (i = 0; i <= 21; i++) {
        *x0 = 0.0;                              //  Set initial states to zero
    }
}
```

```
/* Function: mdlOutputs =======================================================
 * Abstract:
 *      Output blade velocity
 */
static void mdlOutputs(SimStruct *S, int_T tid) {
    /* Inputs, outputs, work vectors and states: */
    real_T            *y            = ssGetOutputPortRealSignal(S,0);
    real_T            *x            = ssGetContStates(S);
    real_T            *r            = ssGetRWork(S);
    InputRealPtrsType uPtrs         = ssGetInputPortRealSignalPtrs(S,0);
    InputRealPtrsType uPtrs1        = ssGetInputPortRealSignalPtrs(S,1);
    InputRealPtrsType uPtrs2        = ssGetInputPortRealSignalPtrs(S,2);

    /* Variable Arrays: */
    real_T            *RHS          = mxGetPr(PARAM_1(S));
    real_T            *MFR150       = mxGetPr(PARAM_3(S));
    real_T            *LDMFR150     = mxGetPr(PARAM_4(S));
    real_T            *LDMFmodel    = mxGetPr(PARAM_5(S));
    real_T            *VDdashbar    = mxGetPr(PARAM_6(S));
    real_T            *VIDbar       = mxGetPr(PARAM_7(S));
    real_T            *T_delta      = mxGetPr(PARAM_8(S));
    real_T            *T_gamma      = mxGetPr(PARAM_9(S));
    real_T            *TgTd         = mxGetPr(PARAM_10(S));
    real_T            *FDAMP        = mxGetPr(PARAM_11(S));
    real_T            *FGC          = mxGetPr(PARAM_12(S));

    /* Fixed Arrays: */
    const real_T      *MFcoeff          = mxGetPr(PARAM_13(S));
    const real_T      *LDMFcoeff        = mxGetPr(PARAM_14(S));
    const real_T      *w                = mxGetPr(PARAM_15(S));
    const real_T      *Ap               = mxGetPr(PARAM_16(S));
    const real_T      *Thetao_deg       = mxGetPr(PARAM_17(S));
    const real_T      *A1_deg           = mxGetPr(PARAM_18(S));
    const real_T      *B1_deg           = mxGetPr(PARAM_19(S));
    const real_T      *VDdash           = mxGetPr(PARAM_20(S));
```

```
const real_T      *VID              = mxGetPr(PARAM_21(S));
const real_T      *dVDdashdtheta    = mxGetPr(PARAM_22(S));
const real_T      *WoDdash          = mxGetPr(PARAM_23(S));
const real_T      *VoDdash          = mxGetPr(PARAM_24(S));
const real_T      *WDdash           = mxGetPr(PARAM_25(S));
const real_T      *Rins             = mxGetPr(PARAM_26(S));
const real_T      *Thetao           = mxGetPr(PARAM_27(S));
const real_T      *ThetaD           = mxGetPr(PARAM_28(S));
const real_T      *A1               = mxGetPr(PARAM_29(S));
const real_T      *B1               = mxGetPr(PARAM_30(S));
const real_T      *XBD              = mxGetPr(PARAM_31(S));
const real_T      *XB               = mxGetPr(PARAM_32(S));
const real_T      *XLDE             = mxGetPr(PARAM_33(S));
const real_T      *TID              = mxGetPr(PARAM_34(S));
const real_T      *WID              = mxGetPr(PARAM_35(S));

/* Run-time Variables: */
int_T             i, j;
real_T            Time, pressure, Modes_S, Modes_E;
real_T            delTheta_deg, velocity, F_pounds, VDAMPX, ans1;
real_T            BETA, ZETA, BETAD, ZETAD, THETA, THETAD, DELTA, GAMMA;
real_T            XDdash, YDdash, ZDdash, LDdash, XBDX, XBDY, XBDZ;
real_T            T1, T2, T3, T4, T5, T6, T7;

Time = U(0);
Modes_S = U1(0)-1;
Modes_E = U1(1)-1;
pressure = U2(0)/Ap[0];

/* Delta pitch angle (deg) */
delTheta_deg = Thetao_deg[0]-A1_deg[0]*cos(w[0]*Time)-B1_deg[0]*sin(w[0]*Time);

for (i = 0; i <= 7; i = i + 1) {
    VDdashbar[i] = VDdash[i];          //  VDdash in deg; VDdashbar in rad
    VIDbar[i] = VID[i];
```

```
}

VDdashbar[2] = VDdash[2] + dVDdashdtheta[2]*delTheta_deg*VDdash[3];
VIDbar[2] = VID[2] + dVDdashdtheta[2]*delTheta_deg*VID[3];

BETA = WoDdash[0];
ZETA = VoDdash[0];
BETAD = 0;
ZETAD = 0;

for (i = Modes_S; i <= Modes_E; i = i + 1) {
    BETA = BETA + x[(2*i)+5]*WDdash[i]*Rins[0];          //  Flap slope (rads)
    ZETA = ZETA + x[(2*i)+5]*VDdashbar[i]*Rins[0];       //  Lag slope (rads)
    // Rate of change (d/d(psi))
    BETAD = BETAD + x[(2*i)+6]*WDdash[i]*Rins[0];        //  Convert to time
    ZETAD = ZETAD + x[(2*i)+6]*VDdashbar[i]*Rins[0];     //  from azimuth
}

/* Pitch angle (rads) */
THETA = Thetao[0] + ThetaD[0] - A1[0]*cos(w[0]*Time) - B1[0]*sin(w[0]*Time);
THETAD = (A1[0]*sin(w[0]*Time)-B1[0]*cos(w[0]*Time))*w[0];

/* Global position of outboard damper attachment D */
XDdash = cos(BETA)*cos(ZETA)*XBD[0] + XB[0];
YDdash = sin(ZETA)*XBD[0] + XB[1];
ZDdash = sin(BETA)*cos(ZETA)*XBD[0] + XB[2];

LDdash = sqrt(((XDdash-XLDE[0])*(XDdash-XLDE[0]))+((YDdash-XLDE[1])*...
        ...(YDdash-XLDE[1]))+((ZDdash-XLDE[2])*(ZDdash-XLDE[2])));

DELTA = atan((ZDdash-XLDE[2])/(XDdash-XLDE[0]));        //  LDdash cancels
GAMMA = acos((sqrt(((XDdash-XLDE[0])*(XDdash-XLDE[0]))+((ZDdash-XLDE[2])*...
        ...(ZDdash-XLDE[2]))))/LDdash);

XBDX = XBD[0];
```

```
XBDY = XBD[1];
XBDZ = XBD[2];

/* Derivation from original R150 code */
ans1 = -sin(DELTA)*sin(ZETA)*cos(BETA)*cos(GAMMA)*...
        ...cos(THETA)*BETAD*XBDY+sin(DELTA)*cos(BETA)*...
        ...cos(GAMMA)*cos(THETA)*THETAD*XBDY+sin(DELTA)*...
        ...cos(BETA)*cos(GAMMA)*cos(ZETA)*BETAD*XBDX+sin(THETA)*...
        ...sin(ZETA)*cos(BETA)*cos(GAMMA)*cos(DELTA)*THETAD*XBDY+...
        ...sin(THETA)*cos(BETA)*cos(GAMMA)*cos(DELTA)*cos(ZETA)*...
        ...ZETAD*XBDZ-sin(THETA)*cos(BETA)*cos(GAMMA)*cos(DELTA)*...
        ...BETAD*XBDY+sin(ZETA)*cos(BETA)*cos(GAMMA)*cos(DELTA)*...
        ...cos(THETA)*THETAD*XBDZ-sin(ZETA)*cos(BETA)*cos(GAMMA)*...
        ...cos(DELTA)*ZETAD*XBDX-cos(BETA)*cos(GAMMA)*cos(DELTA)*...
        ...cos(THETA)*cos(ZETA)*ZETAD*XBDY-cos(BETA)*cos(GAMMA)*...
        ...cos(DELTA)*cos(THETA)*BETAD*XBDZ;
VDAMPX = sin(BETA)*sin(DELTA)*sin(THETA)*sin(ZETA)*cos(GAMMA)*THETAD*...
        ...XBDY+sin(BETA)*sin(DELTA)*sin(THETA)*cos(GAMMA)*cos(ZETA)*...
        ...ZETAD*XBDZ-sin(BETA)*sin(DELTA)*sin(THETA)*cos(GAMMA)*BETAD*...
        ...XBDY+sin(BETA)*sin(DELTA)*sin(ZETA)*cos(GAMMA)*cos(THETA)*...
        ...THETAD*XBDZ-sin(BETA)*sin(DELTA)*sin(ZETA)*cos(GAMMA)*ZETAD*...
        ...XBDX-sin(BETA)*sin(DELTA)*cos(GAMMA)*cos(THETA)*cos(ZETA)*...
        ...ZETAD*XBDY-sin(BETA)*sin(DELTA)*cos(GAMMA)*cos(THETA)*BETAD*...
        ...XBDZ-sin(BETA)*sin(THETA)*sin(ZETA)*cos(GAMMA)*cos(DELTA)*...
        ...BETAD*XBDZ+sin(BETA)*sin(THETA)*cos(GAMMA)*cos(DELTA)*THETAD*...
        ...XBDZ+sin(BETA)*sin(ZETA)*cos(GAMMA)*cos(DELTA)*cos(THETA)*...
        ...BETAD*XBDY-sin(BETA)*cos(GAMMA)*cos(DELTA)*cos(THETA)*THETAD*...
        ...XBDY-sin(BETA)*cos(GAMMA)*cos(DELTA)*cos(ZETA)*BETAD*XBDX+...
        ...sin(GAMMA)*sin(THETA)*sin(ZETA)*ZETAD*XBDZ-sin(GAMMA)*...
        ...sin(THETA)*cos(ZETA)*THETAD*XBDY-sin(GAMMA)*sin(ZETA)*...
        ...cos(THETA)*ZETAD*XBDY-sin(GAMMA)*cos(THETA)*cos(ZETA)*THETAD...
        ...*XBDZ+sin(GAMMA)*cos(ZETA)*ZETAD*XBDX+sin(DELTA)*sin(THETA)*...
        ...sin(ZETA)*cos(BETA)*cos(GAMMA)*BETAD*XBDZ-sin(DELTA)*...
        ...sin(THETA)*cos(BETA)*cos(GAMMA)*THETAD*XBDZ+ans1;
```

```
velocity = (-VDAMPX*2.54)/100;                    //  Velocities are negative
F_pounds = (Ap[0]*pressure*2.205)/9.80665;        //  lbf for MF150

FDAMP[0] = F_pounds;
FDAMP[1] = 0;
FDAMP[2] = 0;

/* Damper translation matrices */
T_delta[0] = cos(DELTA);
T_delta[1] = 0;
T_delta[2] = sin(DELTA);
T_delta[3] = 0;
T_delta[4] = 1;
T_delta[5] = 0;
T_delta[6] = -sin(DELTA);
T_delta[7] = 0;
T_delta[8] = cos(DELTA);

T_gamma[0] = cos(GAMMA);
T_gamma[1] = sin(GAMMA);
T_gamma[2] = 0;
T_gamma[3] = -sin(GAMMA);
T_gamma[4] = cos(GAMMA);
T_gamma[5] = 0;
T_gamma[6] = 0;
T_gamma[7] = 0;
T_gamma[8] = 1;

/* Forces and moments at C due to motion of damper:
 * FGD = T_delta*T_gamma*FDAMP; FGC = FGD;
 */

/* T_delta*T_gamma */
for (i = 0; i <= 2; i = i + 1) {
    for (j = 0; j <= 2; j = j + 1) {
```

```
            TgTd[(i*3)+j] = T_delta[j]*T_gamma[(i*3)] + T_delta[j+3]*...
                ...T_gamma[(i*3)+1] + T_delta[j+6]*T_gamma[(i*3)+2];
        }
    }

    /* TgTd*FDAMP */
    for (i = 0; i <= 2; i = i + 1) {
        FGC[i] = TgTd[i]*FDAMP[0] + TgTd[i+3]*FDAMP[1] + TgTd[i+6]*FDAMP[2];
    }

    /* Calculating forcing by lag damper to put into equations of motion
     * Taken from NLDAM3 bottom section - all linearized for small angles
     */

    for (i = Modes_S; i <= Modes_E; i = i + 1) {

        T1 = FGC[0]*(-THETA*XBD[1] - XBD[2] - XBD[0]*BETA)*WDdash[i];
        T2 = FGC[0]*(THETA*XBD[2] - XBD[1] - XBD[0]*ZETA)*VDdashbar[i];
        T3 = FGC[0]*(-BETA*XBD[1]*ZETA*XBD[2])*TID[i];
        T4 = FGC[1]*VIDbar[i];
        T5 = -FGC[1]*XBD[2]*TID[i];
        T6 = FGC[2]*WID[i];
        T7 = FGC[2]*XBD[1]*TID[i];

        LDMFmodel[i] = (T1+T2+T3+T4+T5+T6+T7)/(w[0]*w[0]*Rins[0]);

        /* Fourier reconstruction to find modal forcings at time T */
        MFR150[i] = MFcoeff[(i*17)];
        LDMFR150[i] = LDMFcoeff[(i*17)];
        for (j = 0; j <= 7; j = j + 1) {
            MFR150[i] = MFR150[i] - MFcoeff[(i*17)+(2*j)+1]*...
                        ...cos((j+1)*w[0]*Time) - MFcoeff[(i*17)+(2*j)+2]*...
                        ...sin((j+1)*w[0]*Time);
            LDMFR150[i] = LDMFR150[i] - LDMFcoeff[(i*17)+(2*j)+1]*...
                        ...cos((j+1)*w[0]*Time) - LDMFcoeff[(i*17)+(2*j)+2]*...
```

```
                        ...sin((j+1)*w[0]*Time);
        }
        RHS[i] = MFR150[i] - LDMFR150[i] + LDMFmodel[i];
    }

    y[0] = velocity;                    //  Output Velocity
    for (i = 0; i <= 7; i = i + 1) {    //  Pass work function to next time step
        r[i] = RHS[i];
    }
}

#define MDL_DERIVATIVES /* Function: mdlDerivatives ============================
 */
 * Abstract:
 *      Calculate derivatives
 */
static void mdlDerivatives(SimStruct *S) {
    /* Inputs, outputs, work vectors and states: */
    real_T            *dx       = ssGetdX(S);
    real_T            *x        = ssGetContStates(S);
    real_T            *r        = ssGetRWork(S);

    /* Variable Arrays: */
    real_T            *RHS_in   = mxGetPr(PARAM_2(S));

    /* Fixed Arrays: */
    const real_T      *w        = mxGetPr(PARAM_15(S));
    const real_T      *I        = mxGetPr(PARAM_36(S));
    const real_T      *lambda   = mxGetPr(PARAM_37(S));
    const real_T      *v        = mxGetPr(PARAM_38(S));
    int_T             i;

    for (i = 0; i <= 7; i = i + 1) {        //  Work function from last time step
        RHS_in[i] = r[i];
    }
```

```
    /* States of Eq. (6.2) */
    dx[0]  = 0;
    dx[1]  = 0;
    dx[2]  = 0;
    dx[3]  = 0;
    dx[4]  = 0;
    dx[5]  = x[6];
    dx[6]  = ((RHS_in[0]/I[0])-(lambda[0]*lambda[0])*x[5]-2*v[0]*...
             ...lambda[0]*x[6]/w[0])*(w[0]*w[0]);
    dx[7]  = x[8];
    dx[8]  = ((RHS_in[1]/I[1])-(lambda[1]*lambda[1])*x[7]-2*v[1]*...
             ...lambda[1]*x[8]/w[0])*(w[0]*w[0]);
    dx[9]  = x[10];
    dx[10] = ((RHS_in[2]/I[2])-(lambda[2]*lambda[2])*x[9]-2*v[2]*...
             ...lambda[2]*x[10]/w[0])*(w[0]*w[0]);
    dx[11] = x[12];
    dx[12] = ((RHS_in[3]/I[3])-(lambda[3]*lambda[3])*x[11]-2*v[3]*...
             ...lambda[3]*x[12]/w[0])*(w[0]*w[0]);
    dx[13] = x[14];
    dx[14] = ((RHS_in[4]/I[4])-(lambda[4]*lambda[4])*x[13]-2*v[4]*...
             ...lambda[4]*x[14]/w[0])*(w[0]*w[0]);
    dx[15] = x[16];
    dx[16] = ((RHS_in[5]/I[5])-(lambda[5]*lambda[5])*x[15]-2*v[5]*...
             ...lambda[5]*x[16]/w[0])*(w[0]*w[0]);
    dx[17] = x[18];
    dx[18] = ((RHS_in[6]/I[6])-(lambda[6]*lambda[6])*x[17]-2*v[6]*...
             ...lambda[6]*x[18]/w[0])*(w[0]*w[0]);
    dx[19] = x[20];
    dx[20] = ((RHS_in[7]/I[7])-(lambda[7]*lambda[7])*x[19]-2*v[7]*...
             ...lambda[7]*x[20]/w[0])*(w[0]*w[0]);

}

/* Function: mdlTerminate =============================================================
```

```
 * Abstract:
 *    No termination needed, but we are required to have this routine.
 */
static void mdlTerminate(SimStruct *S) {
}

#ifdef  MATLAB_MEX_FILE         //  Is this file being compiled as a MEX-file?
#include "simulink.c"           //  MEX-file interface mechanism
#else
#include "cg_sfun.h"            //  Code generation registration function
#endif
```

Bibliography

[1] A. Blakeborough, M.S. Williams, A.P. Darby, and D.M. Williams. The development of real-time substructure testing. *Phil. Trans. R. Soc. Lond. A*, 359:1869–1891, 2001.

[2] R.E. Eyres. *Vibration Reduction In Helicopters Using Lag Dampers.* PhD thesis, University of Bristol, UK, 2005.

[3] A.J. Kappos. *Dynamic Loading and Design of Structures.* Spon Press, London and New York, 2002.

[4] S. P. Timoshenko. *Vibration problems in engineering.* Van Nostrand, 1937.

[5] W. Weaver Jr, S. P. Timoshenko, and D.H. Young. *Vibration problems in engineering.* Wiley, 1990.

[6] R.E.D. Bishop and D.C. Johnson. *The Mechanics of Vibration.* Cambridge University Press, 1960.

[7] R.W. Clough and J. Penzien. *Dynamics of Structures.* McGraw-Hill, Inc., 1993.

[8] M.S. Williams and J.D. Todd. *Structures: theory and analysis.* Macmillan, 2000.

[9] A.K. Chopra. *Dynamics of Structures: Theory and Applications to Earthquake Engineering.* Prentice Hall, Inc., 2001.

[10] M.S. Williams and A. Blakeborough. Laboratory testing of structures under dynamic loads: an introductory review. *Phil. Trans. R. Soc. Lond. A*, 359: 1651 – 1669, 2001.

[11] V.V. Bertero and S. Brokken. Infills in seismic resistant building. *J. Strut. Engng.*, 109(6):1337–1361, 1983.

[12] Han-Seon Lee and Sung-Woo Woo. Effect of masonry infills on seismic performance of a 3-storey r/c frame with non-seismic detailing. *Earthquake Engng Struct. Dynam.*, 31:353–378, 2002.

[13] A.B. Mehrabi, P.B. Shing, M.P. Shuller, and J.L. Noland. Experimental evaluation of masonry-infilled rc frames. *J. Strut. Engng.*, 122(3):228–237, 1996.

[14] G.M. Calvi and G.R. Kingsley. Problems and certainties in the experimental simulation of the seismic response of mdof structures. *Engng Structures*, 18 (3):213–226, 1996.

[15] N. Ogawa, K. Ohtani, T. Katayama, and H. Shibata. Construction of a three-dimensional large-scale shaking table and development of core technology. *Phil. Trans. R. Soc. Lond. A*, 359:1725–1751, 2001.

[16] R.T. Severn. Ecoest - european consortium of earthquake shaking tables - overview. *Proc. 10th European conference on earthquake engineering, Viena, Austria*, 4:2987–2992, 1996.

[17] A.J. Crewe. *The characterisation and optimisation of earthquake shaking table performance.* PhD thesis, University of Bristol, UK, 1998.

[18] I.D. Landau. *Adaptive control: the model reference approach.* New York: Marcel Dekker, 1979.

[19] D.P. Stoten and H. Benchoubane. Empirical studies of an mrac algorithm with minimal controller synthesis. *Int. J. Control*, 51:823–849, 1990.

[20] D.P. Stoten and H. Benchoubane. Robustness of a minimal controller synthesis algorithm. *Int. J. Control*, 51:851–861, 1990.

[21] D.P. Stoten and E.G. Gómez. Adaptive control of shaking tables using the minimal control synthesis algorithm. *Phil. Trans. R. Soc. Lond. A*, 359:1697–1723, 2001.

[22] E.G. Gómez. *Application of the MCS algorithm to the control system of the Bristol shaking table.* PhD thesis, University of Bristol, UK, 1999.

[23] R. Žarnić, S. Gostič, A.J. Crewe, and C.A. Taylor. Shaking table tests of 1:4 reduced-scale models of masonry infilled reinforced concrete frame buildings. *Earthquake Engng Struct. Dynam.*, 30:819–834, 2001.

[24] M. Nakashima. Development, potential, and limitations of real-time online (pseudo dynamic) testing. *Phil. Trans. R. Soc. Lond. A*, 359(1786):1851–1867, 2001.

[25] S.A. Mahin, P.B. Shing, C.R. Thewalt, and R.D. Hanson. Pseudodynamictest method current status and future directions. *J. Strut. Engng.*, 115:2113–2128, 1989.

[26] C.K. Shield, C.W. French, and J. Timm. Development and implementation of the effective force testing method for seismic simulation of large-scale structures. *Phil. Trans. R. Soc. Lond. A*, 359(1786):1911–1929, 2001.

[27] J. Zhao, C. French, C. Shield, and T. Posbergh. Considerations for the development of real-time dynamic testing. *Earthquake Engng Struct. Dyn.*, 32(11):1773–1794, 2003.

[28] S.J. Dyke, B.F. Spencer, P. Quast, and M.K. Sain. Role of control-structure interaction in protective system design. *J. Engng. Mech.*, 121(2):322–338, 1995.

[29] J. Dimig, C. Shield, C. French, F. Bailey, and A. Clark. Effective force testing: A method of seismic simulation for structural testing. *J. Strut. Engng.*, 125: 1028–1037, 1999.

[30] J. Zhao, C. French, C. Shield, and T. Posbergh. Nonlinear system modeling and velocity feedback compensation for eft. *J. Engng. Mech.*, 131(3):244–253, 2003.

[31] M. Hakuno, M. Shidawara, and T. Hara. Dynamic destructive test of a cantilever beam controlled by an analog-computer. *Trans. Jpn Soc. Civil Eng.*, 171:1–9, 1969. (In Japanese.).

[32] K. Takanashi, K. Udagawa, M. Seki, T. Okada, and H. Tanaka. Nonlinear earthquake response analysis of structures by a computer actuator online system. (part 1: details of the system.). *Trans. Architectural Inst. Jpn*, 229: 77–83, 1975. (In Japanese.).

[33] S.A. Mahin and P.B. Shing. Pseudodynamic method of seismic testing. *J. Strut. Engng.*, 111:1482–1503, 1985.

[34] S.A. Mahin and P.B. Shing. Computational aspects of a seismic performance test method using on-line computer control. *Earthquake Engng Struct. Dyn.*, 13:507–526, 1985.

[35] S.A. Mahin and P.B. Shing. Cumulative experimental errors in pseudodynamic tests. *Earthquake Engng Struct. Dyn.*, 15:409–424, 1987.

[36] P.B. Shing and S.A. Mahin. Cumulative experimental errors in pseudodynamic tests. *Earthquake Engng Struct. Dynam.*, 15:409–424, 1987.

[37] K. Takanashi and M. Nakashima. Japanese activities on online testing. *J. Engng Mech.*, 113:1014–1032, 1987.

[38] P.B. Shing, M. Nakashima, and O.S. Bursi. Application of pseudodynamic test method to structural research. *Earthquake Spectra*, 12:29–56, 1996.

[39] C.R. Thewalt and S.A. Mahin. Non-planar pseudodynamic testing. *Earthquake Engng Struct. Dyn.*, 24:733–746, 1995.

[40] F.J. Molina, S. Sorace, G. Terenzi, G. Magonette, and B. Viaccoz. Seismic tests on reinforced concrete and steel frames retrofitted with dissipative braces. *Earthquake Engng Struct. Dyn.*, 33:1373–1394, 2004.

[41] G. Magonette. Development and application of large-scale continuous pseudodynamic testing techniques. *Proc. R Soc. A*, 359(1786):1771–1799, 2001.

[42] F.J. Molina, G. Verzeletti, G. Magonette, P. Buchet, and M. Geradin. Bidirectioal pseudodynamic test of a full-siz three-story building. *Earthquake Engng Struct. Dyn.*, 28:1541–1566, 1999.

[43] F.J. Molina, G. Verzeletti, G. Magonette, P. Buchet, V. Renda, M. Geradin, A. Parducci, M. Mezzi, A. Pacchiarotti, L. Federici, and Mascelloni S. Pseudodynamic tests on rubber base isolators with numerical substructuring of the superstructure and strain-rate effect compensation. *Earthquake Engng Struct. Dyn.*, 31(8):1563–1582, 2002.

[44] J. Donea, P. Magonette, P. Negro, P. Pegon, A. Pinto, and G. Verzeletti. Pseudodynamic capabilities of the ELSA laboratory for earthquake testing of large structures. *Earthquake Spectra*, 12(1):163–180, 1996.

[45] Yu-Yuan Lin, Kuo-Chun Chang, and Yuan-Li Wang. Comparison of displacement coefficient method and capacity spectrum method with experimental results of RC columns. *Earthquake Engng Struct. Dynam.*, 33 (1):35–48, 2004.

[46] P. Pan, M. Nakashima, and H. Tomofuji. Online test using displacement-force mixed control. *Earthquake Engng Struct. Dynam.*, 34(8):869–888, 2005.

[47] F. Casciati and G. Magonette. Testing facilities and laboratory validation. *Proc. Advance in structral control, CINME, Barcelona, Spain*, pages 1–23, 1999.

[48] M. Nakashima and N. Masaoka. Real-time on-line test for MDOF systems. *Earthquake Engng Struct. Dynam.*, 28:393–420, 1999.

[49] C.R. Thewalt and S.A. Mahin. Unconditionally stabile hybrid pseudodynamic algorithm. *Earthquake Engng Struct. Dyn.*, 24:723–731, 1995.

[50] Y. Zhang, R. Sause, J.M. Ricles, and C.J Naito. Modified predictorcorrector numerical scheme for real-time pseudo dynamic tests using state-space formulation. *Earthquake Engng Struct. Dyn.*, 34:271–288, 2005.

[51] Yen-Po Wang, Chien-Liang Lee, and Tzen-Hun Yo. Modified state-space procedures for pseudodynamic testing. *Earthquake Engng Struct. Dyn.*, 30: 59–80, 2001.

[52] M. Nakashima, T. Kaminosono, M. Ishida, and K. Ando. Integration techniques for substructure pseudodynamic test. *Proc. 4th U.S. National Conference on Earthquake Engineering, Palm Springs, CA*, pages 515–524, 1990.

[53] S.Y. Chang, K.C. Tsai, and K.C. Chen. Improved time integration for pseudodynamic tests. *Earthquake Engng Struct. Dynam.*, 27:117–130, 1998.

[54] F. Lopez-Almansa, A.H. Harbat, and J. Rodellar. Ssp algorithm for linear and nonlinear dynamic response simulation. *Int. J. Numerical Methods in Engng*, 26:2687–2706, 1988.

[55] O.S. Bursi and P.B. Shing. Evaluation of some implicit time-stepping algorithms for pseudodynamic tests. *Earthquake Engng Struct. Dynam.*, 25 (4):333–355, 1996.

[56] A. Bonelli and O.S. Bursi. Generalized-α methods for seismic structural testing. *Earthquake Engng Struct. Dynam.*, 33(10):1067–1102, 2004.

[57] D. Combescure and P Pegon. α-operator splitting time integration technique

for pseudodynamic testing: error propagation analysis. *Soil Dyn. Earthquake Engng*, 16:427–443, 1997.

[58] N.M. Newmark. A method of computation for structural dynamics. *J. Engng. Mech.*, 85(EM3):67–94, 1959.

[59] H.M. Hilber, T.J.R. Hughes, and R.L. Taylor. Improved numerical dissipation for time integration in structural dynamics. *Earthquake Engng Struct. Dynam.*, 5:283–292, 1977.

[60] M. Nakashima, H. Kato, and E. Takaoka. Development of real-time pseudo dynamic testing. *Earthquake Engng Struct. Dynam.*, 21:779–92, 1992.

[61] T. Horiuchi, M. Inoue, T. Konno, and Y. Namita. Real-time hybrid experimental system with actuator delay compensation and its application to a piping system with energy absorber. *Earthquake Engng Struct. Dynam.*, 28:1121–1141, 1999.

[62] T. Horiuchi and T. Konno. A new method for compensating actuator delay in real-time hybrid experiments. *Phil. Trans. R. Soc. Lond. A*, 359:1893 - 1909, 2001.

[63] S.N. Dermitzakis and S.A Mahin. Development of substructuring techniques for on-line computer controlled seismic performance testing. *Report UBC/EERC-85/04*, 1985.

[64] A.V. Pinto, P. Pegon, G. Magonette, and G. Tsionis. Pseudo-dynamic testing of bridges using non-linear substructuring. *Earthquake Engng Struct. Dynam.*, 33:1125–1146, 2004.

[65] P. Pegon and A. V. Pinto. Pseudo-dynamic testing with substructuring at the elsa laboratory. *Earthquake Engng Struct. Dyn.*, 29:905–925, 2000.

[66] A.P. Darby, A. Blakeborough, and M.S. Williams. Real-time substructure tests using hydraulic actuator. *J. Engng. Mech.*, 125(10):1133–1139, 1999.

[67] D.J. Wagg and D.P. Stoten. Substructuring of dynamical systems via the adaptive minimal control synthesis algorithm. *Earthquake Engng Struct. Dynam.*, 30:865–877, 2001.

[68] M.V. Sivaselvan, A. Reinhorn, Z. Liang, and X. Shao. Real-time dynamic hybrid testing of structural systems. *Proc. 13th World Conf. Earthquake Engineering*, 2004.

[69] M.I. Wallace, S.A. Neild, J. Sieber, D.J. Wagg, and B. Krauskopf. A delay differential equation approach to real-time dynamic substructuring. *Earthquake Engng Struct. Dynam.*, 34(15):1817–1832, 2005.

[70] M.I. Wallace, D.J. Wagg, and S.A. Neild. Multi-actuator substructure testing with applications to earthquake engineering: how do we assess accuracy? *Proc. 13th World Conf. Earthquake Engineering, Vancouver, Canada, 1-6 August*, pages 1–14, 2004. Paper No. 3241.

[71] S.A. Neild, D.P. Stoten, D. Drury, and D.J. Wagg. Control issues relating to real-time substructuring experiments using a shaking table. *Earthquake Engng Struct. Dynam.*, 34(9):1171–1192, 2005.

[72] C.N. Lim, S.A. Neild, D.P. Stoten, D. Drury, and C.A. Taylor. Adaptive control strategy for dynamic substructuring tests. *J. Engng. Mech.*, 2005.

[73] A.P. Darby, M.S. Williams, and A. Blakeborough. Stability and delay compensation for real-time substructure testing. *J. Engng. Mech.*, 128(12): 1276–1284, 2002.

[74] P.J. Gawthrop, M.I. Wallace, S.A. Neild, and D.J. Wagg. Robust real-time substructuring techniques for under damped systems. *Struct. Control and Health Monitoring*, 2005. Submitted.

[75] M.I. Wallace, D.J. Wagg, and S.A. Neild. An adaptive polynomial based forward prediction algorithm for multi-actuator real-time dynamic substructuring. *Proc. R Soc. A*, 461(2064):3807–3826, 2005.

[76] P.J. Gawthrop, M.I. Wallace, and D.J. Wagg. Bond-graph based substructuring of dynamical systems. *Earthquake Engng Struct. Dynam.*, 34 (6):687–703, 2005.

[77] S.A. Neild, D. Drury, D.J. Wagg, D.P. Stoten, and A.J. Crewe. Implementing real time adaptive control methods for substructuring of large structures. *Proc. 3rd World Conf. on Structural Control*, 2002.

[78] C.N. Lim, S.A. Neild, D.P. Stoten, C.A. Taylor, and D. Drury. Using adaptive control for dynamic substructuring tests. *Proc. 3rd European Conf. on Structural Control*, 2004.

[79] B. Wu, H. Bao, J. Ou, and S. Tian. Stability and accuracy analysis of the central difference method for real-time substructure testing. *Earthquake Engng Struct. Dyn.*, 34(7):705–718, 2005.

[80] B. Wu, G. Xu, Q. Wang, and M.S. Williams. Operator-splitting method for real-time substructure testing. *Earthquake Engng Struct. Dynam.*, 2001. In Press.

[81] G. Mosqueda. *Continuous hybrid simulation with geographically distributed substructures.* PhD thesis, University of California, Berkeley, USA, 2003.

[82] A.P. Darby, A. Blakeborough, and M.S. Williams. Improved control algorithm for real-time substructure testing. *Earthquake Engng Struct. Dynam.*, 30:431–448, 2001.

[83] N. Yamamura and H. Tanaka. Response analysis of flexible mdf systems for multiple-support seismic excitations. *Earthquake Engng Struct. Dynam.*, 19: 345–357, 1990.

[84] P. Eykhoff. *System Identification: Parameter and State Estimation.* John Wiley & Sons, 1974.

[85] R.C Dorf and R.H Bishop. *Modern Control Systems.* Number 10. Pearson Prentice Hall, Inc., New Jersy, U.S.A., 2005.

[86] L. Ljung and T. Glad. *Modeling of Dynamic Systems.* Prentice Hall, Inc., 1994.

[87] F.E. Cellier. *Continuous system modelling.* Springer-Verlag, 1991.

[88] D. Karnopp, D.L. Margolis, and R.C. Rosenberg. *System Dynamics : Modeling and Simulation of Mechatronic Systems.* Number 3. Horizon Publishers and Distributors Inc., 2000.

[89] P.J. Gawthrop and S. Scavarda. Special issue on bond graphs: Editorial. *Proc. Institution of Mech. Engineers Pt. I*, 216(I1):i–v, 2002.

[90] G.C. Buttazzo. *Hard Real-Time Computing Systems : Predictable Scheduling Algorithms and Applications (Real-Time Systems Series)*, volume 2. Springer Science and Business Media Inc., 2005.

[91] P. Ashwin. Non-linear dynamics, loss of synchronization and symmetry breaking. *Proc. Institution of Mech. Engineers Pt. G*, 212(3):183–187, 1998.

[92] O. Diekmann, S. van Gils, S.M. Verduyn Lunel, and H.O. Walther. *Delay Equations*, volume 110. Appl. Math. Sciences, 1995.

[93] G. Stépan. *Retarded Dynamical Systems: Stability and Characteristic Functions.* Longman Scientific & Technical, 1989.

[94] K. Engelborghs, T. Luzyanina, and D. Roose. Numerical bifurcation analysis of delay differential equations using DDE-BIFTOOL. *ACM Trans. Math. Software*, 28(1):1–21, 2002.

[95] T. Kalmar-Nagy, G. Stepan, and F.C. Moon. Subcritical Hopf bifurcation in the time delay equation model for machine tool vibration. *Nonlinear Dynamics*, 26:121–142, 2001.

[96] D.E. Gilsinn. Estimating critical Hopf bifurcation parameters for a second order delay differential equation with application to machine tool chatter. *Nonlinear Dynamics*, 30:103–154, 2002.

[97] L. Larger and J. Goedgebuer. Subcritical Hopf bifurcation in dynamical systems described by a scalar nonlinear delay differential equation. *Physical Review E*, 69(036210):1–5, 2004.

[98] B. Cahlon and D. Schmidt. Stability criteria for certain second-order delay differential equations with mixed coefficients. *J. Computational and Appl. Math.*, 170:79–102, 2004.

[99] G.C. Goodwin, S.F. Graebe, and M.E. Salgado. *Control System Design.* Prentice Hall, Inc., 2001.

[100] E. Kreyszig. *Advanced Engineering Mathematics, 8^{th} Edition.* New York: Wiley, 1999.

[101] K.J. Åström and B. Wittenmark. *Adaptive Control.* Addison Wesley, 1995.

[102] P.J. Gawthrop, Ballance D.J., and Vink D. Bond graph based control using virtual actuators. *Proc. 13th European Simulation Symposium: Simulation in Industry, Marseille, France, October*, pages 813–817, 2001.

[103] P.J. Gawthrop. Bond graph based control using virtual actuators. *Proc. Institution of Mech. Engineers Pt. I*, 218(4):251–268, 2004.

[104] P.C. Sen. *Principles of Electrical Machines and Power Electronics.* Number 2. John Wiley & Sons, 1997.

[105] L. Wang and W.R. Cluett. *From Plant Data to Process Control.* Taylor and Francis, London and New York, 2000.

[106] P.J. Gawthrop. Physically-plausible models for identification. *Proc. International Conference On Bond Graph Modeling and Simulation (ICBGM'03), Simulation Series, Orlando, Florida, U.S.A., January*, 2003.

[107] P.J. Gawthrop. Sensitivity bond graphs. *J. Franklin Institute*, 337(7):907–922, 2000.

[108] R.F. Ngwompo and P.J. Gawthrop. Bond graph based simulation of nonlinear inverse systems using physical performance specifications. *J. Franklin Institute*, 336(8):1225–1247, 1999.

[109] R.F. Ngwompo, S. Scavarda, and D. Thomasset. Physical model-based inversion in control systems design using bond graph representation part 1: theory. *Proc. Institution of Mech. Engineers Pt. I*, 215(2):95–103, 2001.

[110] R.F. Ngwompo, S. Scavarda, and D. Thomasset. Physical model-based inversion in control systems design using bond graph representation part 2: applications. *Proc. Institution of Mech. Engineers Pt. I*, 215(2):105–112, 2001.

[111] W. Johnson. *Helicopter Theory.* Princeton University Press, 1980.

[112] I. Chopra. Perspectives in aeromechanical stability of helicopter rotors. *Vertica*, 14(4):457–508, 1990.

[113] H. Kang. *Rotor blade lag damping using embedded chordwise absorbers.* PhD thesis, The Pennsylvania State University, USA, 2001.

[114] J.C. Houbolt and G.W. Brooks. Differential equations of motion for combined flapwise bending, chordwise bending, and torsion of twisted nonuniform rotor blades. *Technical Report, NASA*, 1346, 1957.

[115] G. Isakson and J.G. Eisley. Natural frequencies in coupled bending and torsion of twisted rotating and nonrotating blades. *Technical Report, NASA*, CR-65, 1964.

[116] C. Young. A method of predicting the loading on helicopter rotor blades. *Technical Report, Royal Aircraft Establishment*, 82096, 1982.

[117] R.E. Eyres, A.R. Champneys, and N.A.J. Lieven. Modelling and dynamic response of a damper with relief valve. *Nonlinear Dynamics*, 40:119–147, 2005.

[118] M.I. Wallace, D.J. Wagg, and S.A. Neild. Use of control techniques for error analysis of real time dynamic substructure testing. *Proc. 3rd European Conf. on Structural Control, Vienna, Austria, 12-15 July*, pages 59–62, 2004.

[119] A. Gonzalez-Buelga, D.J. Wagg, M.I. Wallace, S.A. Neild, and J.H.G. Macdonald. Testing an autoparametric pendulum using a hybrid numerical experimental method. *Proc. 1st Int. Conf. on Advances in Experimental Structural Engineering, Nagoya, Japan, 19-21 July*, pages 417–424, 2005.

[120] M.I. Wallace, A. Gonzalez-Buelga, S.A. Neild, and D.J. Wagg. Control techniques for real-time dynamic substructuring. *Proc. IADAT Int. Conf. on Automation, Control and Instrumentation, Bilbao, Spain, 2-4 February*, pages 56–60, 2005.

[121] M.I. Wallace, S.A. Neild, and D.J. Wagg. A stability analysis of substructured systems for real-time dynamic testing. *Proc. 1st Int. Conf. on Advances in Experimental Structural Engineering, Nagoya, Japan, 19-21 July*, pages 385–392, 2005.

[122] M.I. Wallace, S.A. Neild, J. Sieber, D.J. Wagg, and B. Krauskopf. Delay differential equation models for real-time dynamic substructuring. *Proc. IDETC/CIE ASME International Design Engineering Technical Conferences, California, USA, 24-28 September*, pages 1–8, 2005. Paper No. 85010.